OBPP

OBSOLETE BOOKS
PRESERVATION
PROJECT

Omslag: Rong van Engelen

I.S.B.N.: 90 - 70556 - 19 - 7

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Engelen, Wichert van

Op safari door de Atari : het complete Atari boek voor de 400/800/600XL/800XL/65XE/130XE / Wichert van Engelen ;
[ill. van de auteur]. - Amsterdam : Wolfkamp. - Ill.
Met index, reg.
ISBN 90-70556-19-7
SISO 365.3 UDC 681.3.06
Trefw.: Atari (computer) / programmeren (computer).

Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt, op welke wijze dan ook, zonder voor- afgaande schriftelijke toestemming van de uitgever.

Een uitgave van: Uitgeverij WOLFKAMP
Weteringschans 221
Amsterdam

© 1985, Uitgeverij WOLFKAMP

OP SAFARI DOOR DE ATARI

het complete ATARI boek voor de
400/800/600XL/800XL/65XE/130XE

WICHERT VAN ENGELEN

Uitgeverij WOLFKAMP. Amsterdam

INHOUDSOPGAVE

VOORWOORD *blz. 8*

1. INLEIDING *blz. 9*
programma schrijven, communicatie mens-computer

2. DE ATARI COMPUTERS *blz. 11*
binnenkant, CPU, het geheugen ROM en RAM, I/O, video aansluitingen, bediening, testen, toetsenbord opmaaktoetsen, speciale toetsen, funktietoetsen internationale letters, schermopmaak, diskdrive

3. DIREKTE OPDRACHTEN *blz. 29*
inleiding, fouten verbeteren, opdrachten

4. EEN PROGRAMMA SCHRIJVEN *blz. 35*
een programma, hulp bij het maken van programma's een programma maken, bewaren en inlezen, disk gebruikers een lijst van alle programma's, het gradenprogramma

5. VARIABELEN *blz. 47*

6. INVOER EN BEELDWEERGAVE *blz. 53*
input, berichten, weergave op het beeldscherm tabulatie, andere schermen, tabuleren in twee richtingen hengelen

7. LUSSEN *blz. 63*
inleiding, tijd meten, terug naar lussen lussen binnen lussen

8. LOGICA EN IF... THEN... *blz. 73*
vergelijken, verschillende mogelijkheden, logische operators, gebruik

9. *STRINGS blz. 81*
wat is een string, stringvariabelen, rekenen met strings en getallen, rekenen met strings, vergelijken van strings palindromen

10. LIJSTEN MET GEGEVENS *blz. 93*
inleiding, data-lijsten, arrays, string arrays de energierekening

11. SORTEREN *blz. 105*
sorteerprogramma, bubblesort, allessorteerder

12. BESTANDEN VERWERKEN *blz. 111*
het nut van bestanden, cassette versus disk maken van bestanden, lezen van een bestand

13. UITSTAPJES *blz. 133*
inleiding, subroutines, kiezen tussen subroutines

14. DE SCHERMEN *blz. 141*
de schermmodi, ruitjespapier, tekstscherm 0, grafische symbolen, de kleur van scherm 0, tekstschermen 1, 2, 12 en 13 verschil modus 0 en modi 1 en 2, kleuren in modi 1 en 2 schermen 12 en 13, de vier-kleuren schermen 3, 5, 7 en 15 grafische opdrachten, kleuren in vier-kleuren modi twee-kleuren schermen 4, 6 en 14, kleuren, twee kleuren scherm 8 geen tekstvenster en wissen van scherm, de GTIA schermen 9, 10 en 11, scherm 9, scherm 10, scherm 11.

15. GRAFIEK EN TEKST *blz. 167*
inleiding, lift-off, lucifers, zelf karakters maken, verplaatsen van de lettertekens, samenstellen van lettertekens, Japans, de lijst in de war, een aap, de woeste rivier

16. GRAFIEK EN KLEUR *blz. 187*
inleiding, assenstelsel, sinusgrafieken, taartgrafiek, voort- schrijdend gemiddelde

17. TOETSENBORD EN JOYSTICKS *blz. 195*
inleiding, toetsenbord, joysticks, raketspel

18. GRAFIEK: HOGE RESOLUTIE *blz. 205*
stippen, lijnen, krommen, cirkels en aanverwanten, inkleuren

19. PLAYER/MISSILE GRAFIEK *blz. 219*
inleiding, maken van een speler, maken van een projectiel, instellen van pm grafiek, enkele lijn resolutie, dubbelgroot en beweging, meer dan een kleur

20. GELUID EN MUZIEK *blz. 235*
geluid, de BASIC besturing, muziek, omhullenden, meer dan een kanaal, geluidseffecten, safarigeluiden

21. DE DERDE DIMENSIE *blz.* ~~*237*~~ 247
een, twee of drie, dieptesuggestie door schaduwwerking, hole in one, tunnels, weergave op het beeldscherm, matrices, projectie, rotatie, een wijnglas, een kubus, een piramide

BIJLAGEN:

1. ATASCII CODES *blz. 272*
2. FOUTMELDINGEN *blz. 277*
3. BASIC STATEMENTS *blz. 283*
4. BELANGRIJKE PEEK EN POKE ADRESSEN *blz. 323*
5. IOCB *blz. 327*
6. DOS *blz. 333*
7. EXTRA GEHEUGEN VAN 130XE *blz. 347*
8. TOETSENBORD CODES *blz. 351*
9. BINAIR/DECIMAAL GETALSTELSEL *blz. 353*

INDEX *blz. 357*

VOORWOORD

Ga mee op een ontdekkingsreis door de wondere wereld van Atari-BASIC. Bekijk het binnenste van de computer, volg de paden van een stroomdiagram en de slingerende sporen van de strings, zie de variabelen veranderen, loop door een woud aan programmatips, rij door het grafische landschap, huiver door de realistische geluidseffecten, geef direkte opdrachten aan uw Atari-hulpje en laat de subroutines het zware werk doen. Kortom, ga op safari door de Atari.
Dit boek is een complete handleiding bij het gebruik van de Atari. Naast een uitleg van de werking en bediening van de Atari en een volledige curus in BASIC, is extra veel aandacht besteed aan de ruime grafische mogelijkheden van de Atari.
Speciaal voor hen die geen grote kennis van wiskunde hebben, is een hoofdstuk over het werken met variabelen opgenomen. Vele programma's met uitleg zetten de lezer aan tot het zelf uitproberen van de vele aspecten van zijn of haar computer. Door het hele boek heen vindt u tips en fijne kneepjes, waardoor de lezer meer uit de computer kan halen.
Grote delen van dit boek hadden niet geschreven kunnen worden zonder de hulp van anderen. Van hen wil ik speciaal bedanken:
Richard Keijzer voor de hulp bij het Japans,
Ellie Rhebergen voor de didactische en wiskundige inbreng,
Janneke Hulscher en *Rong van Engelen* voor de kritische noten.

Veel plezier en succes met het programmeren.

1. INLEIDING

Een programma schrijven

Reeds bij uw eerste kontakten met de wereld der microcomputers heeft u velen horen praten over de aanwezigheid van allerlei schitterende computer-programma's. Een van de redenen om nu juist een **Atari-computer** te kopen, is de enorme hoeveelheid programma's die voor deze computers beschikbaar is. Daarnaast is de Atari, mede door zijn uitstekende grafische mogelijkheden, geschikt voor het zelf maken van programma's.
Maar wat wil dat zeggen: "een programma schrijven"? Waarom zijn er boekenkasten over vol geschreven? Waarom zouden we het zelf moeten doen? Er zijn toch allerlei cassettes en cartridges in de handel met de meest uiteenlopende programma's en spelletjes. Die besparen ons veel intikwerk en nadenken is nauwelijks nodig.
Hier heeft u een van de beste redenen om zelf te gaan (leren) programmeren. **Nadenken.** Als u meer wilt weten over wat computers zijn en wat ze al dan niet kunnen, kunt u niet volstaan met het in de cassetterecorder stoppen van een cassette met weer een spel of nog een boekhoudprogramma.
Maar zelfs als u vindt dat het eigenlijk zonde van de tijd is om te leren programmeren, dan zult u al snel merken dat u er wel iets van af moet weten. De computerwetenschap is nog niet zover gevorderd dat alle programma's echt helemaal op maat zijn. Als u een kant-en-klaar programma optimaal wilt gebruiken, zult u ofwel uw eisen moeten aanpassen, ofwel het programma moeten aanpassen. Daarnaast is voor de werking van veel programma's onontbeerlijk dat de gebruiker iets begrijpt van de werking van de computer. Zelf programma's schrijven voor uw Atari-computer is de beste methode om vertrouwd te raken met de computer en om te begrijpen hoe uw Atari werkt. In dit boek leert u aan de hand van veel oefening hoe de Atari-computers werken en wat u ermee kunt doen. U leert hoe u een probleem kunt vertalen in computertaal en hoe u problemen die te complex of te vaag zijn voor een computer, kunt verdelen in verschillende kleine en minder vage problemen. Bij het schrijven van dit boek is gebruik gemaakt van de **Atari 130XE**. Omdat Atari er altijd voor gewaakt heeft dat de BASIC-taal die door de verschillende Atari-types gebruikt wordt, uitwisselbaar is, kan dit boek ook gebruikt worden voor de ***Atari 400, 600, 600XL, 800XL, 65XE en 130XE***. Bijna alle programma's zullen op elk van deze computers werken. De bezitters van een kleinere Atari zullen soms niet genoeg hebben aan de geheugenruimte in de computer. Een geheugenuitbreiding kan dan (en ook voor vele andere doeleinden) een oplossing zijn. Eigenaars van een 400 of 600 moeten een BASIC-cartridge in de Atari steken om dit boek te kunnen gebruiken.
De programma's in dit boek zijn meestal eenvoudig. U zult ze snel kunnen begrijpen. De programma's in de laatste hoofdstukken veronderstellen de voorafgaande stof als bekend en zijn dus ingewikkelder. Hoewel de programma's allemaal doen wat ze moeten doen, zijn de meeste programma's niet 'af'. In elk programma is altijd plaats voor verbeteringen, of aanpassingen aan de

persoonlijke smaak van de gebruiker. Door de begeleidende tekst bij elk programma zal duidelijk gemaakt worden hoe elk programma is opgebouwd, en van welke tips en truuks gebruik is gemaakt. Daardoor zal de lezer zelf eventuele aanpassingen snel kunnen aanbrengen.

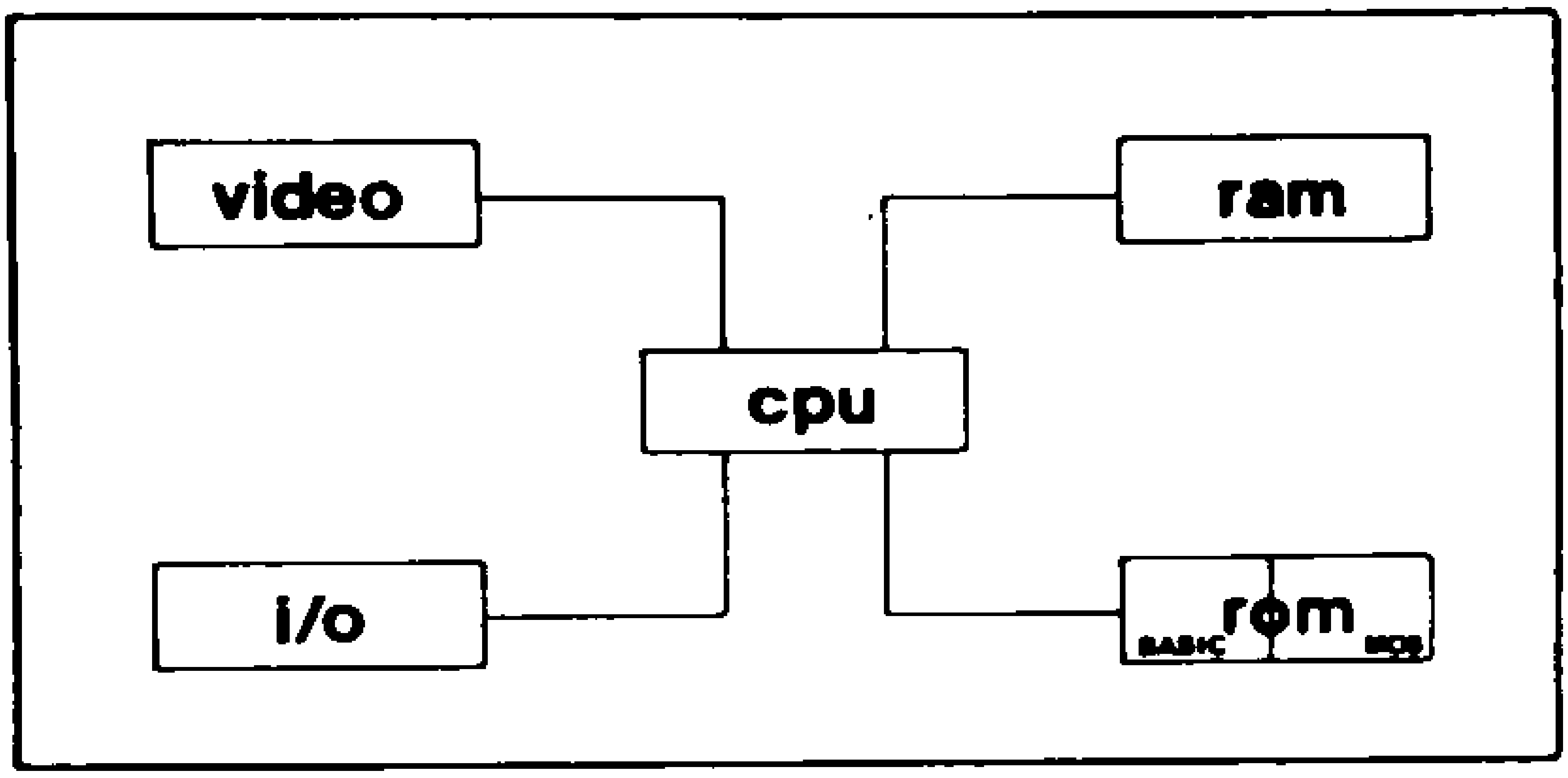
Communicatie mens-computer

De taal waarin we de computer toespreken heet: BASIC. Dit staat voor Beginners All-purpose Symbolic Instruction Code (voor alle doeleinden geschikte symbolische instruktiecode voor beginners). Deze taal werd in de 60-er jaren ontwikkeld in Amerika. Sindsdien zijn er verschillende dialecten ontwikkeld. Atari-BASIC is een van de dialecten. Waarschijnlijk niet een van de meest uitgebreide dialecten, maar daar staat tegenover de gebruiker de beschikking heeft over een indrukwekkend kleurenpallet om grafische afbeeldingen te maken.
Naast BASIC bestaan er nog vele andere computertalen die elk vooral geschikt zijn voor gebruik door gevorderden en een speciale toepassing hebben voor de computer: PASCAL (wiskundige doeleinden), FORTRAN (natuurkundige doeleinden), COBOL (administratieve doeleinden), LISP (kunstmatige intelligentie), FORTH (procesbesturing). Daarnaast zijn er computertalen ontwikkeld die speciaal door kinderen gebruikt kunnen worden. De belangrijkste daarvan zijn LOGO en PILOT.
Voor de Atari-computers zijn op het ogenblik de volgende talen verkrijgbaar: Atari BASIC en Atari Microsoft BASIC, BASIC A+, Forth, C, Pascal, Pilot, Lisp en Machinetaal.
Met BASIC kunt u alle kanten op. Soms gaat het programmeren in BASIC iets omslachtiger, of iets minder elegant dan bij een van de andere talen, maar het voordeel van de eenvoud die BASIC kenmerkt is zo groot, dat zo goed als alle microcomputers standaard met BASIC geleverd worden.

2. DE ATARI COMPUTERS

De binnenkant

Hoewel er verschillen zijn aan te wijzen tussen de verschillende Atari-computers zijn ze allemaal opgebouwd volgens het onderstaande basisprincipe:



sterk vereenvoudigd schema van een Atari-computer

De CPU

De CPU (Central Processing Unit = centrale verwerkingseenheid) is het onderdeel dat centraal in de computer staat. Het bepaalt waar welke gegevens naar toe gaan. Op deze manier 'verwerkt' de CPU de gegevens. Elk gegeven dat de CPU binnenkomt wordt bekeken, waarna de CPU bepaalt wat er verder mee gebeurt. De CPU zou vergeleken kunnen worden met het hoofd van een groot organisatiebureau tijdens een door hen georganiseerd evenement. De CPU van elke Atari-computer is een *6502-microprocessor*.

Dit is wellicht de beste plaats om een veel voorkomende spraakverwarring te verhelpen. Veel mensen maken geen onderscheid tussen een microcomputer en een microprocessor. Een microprocessor is het belangrijkste onderdeel van een microcomputer. Een microcomputer is een machine zoals de Atari. Dat wil zeggen een samenraapsel van een toetsenbord, een kastje, wat chips, enkele andere electronische onderdelen, wat draden en diversen. Deze microcomputer is in staat een heleboel opdrachten die de gebruiker geeft uit te voeren en de resultaten ervan weer te geven via een draad op een beeldscherm, een printer, een cassetterecorder, diskdrive of andere randapparatuur.
Een microprocessor is het centrale en typerende onderdeel van een microcom-

puter. Het vormt de CPU. Er zijn op het ogenblik vele merken en soorten microcomputers op de markt, maar slechts enkele soorten microprocessors. Zodoende hebben heel verschillende computers dezelfde microprocessors.

Het geheugen: ROM en RAM

Het geheugen van een microcomputer bestaat uit een deel **RAM** en een deel **ROM**. RAM staat voor Random Access Memory (=willekeurig toegankelijk geheugen: het vrij beschikbare geheugen). ROM staat voor Read Only Memory (=enkel lees geheugen). De RAM is dat gedeelte van het geheugen waarover de gebruiker vrij kan beschikken om gegevens (waaronder programma's) op te slaan. Niet elke Atari-computer heeft evenveel RAM-geheugen. Van de verschillende computers is het geheugen uit de te breiden door een speciale geheugen-cartridge. De computer die gebruikt is voor dit boek, de *Atari 130XE*, heeft maar liefst 129 K byte geheugen (ROM en RAM samen).
Een K byte is de eenheid waarin een hoeveelheid geheugenruimte wordt aangegeven. Een K byte is een kilo-byte en staat voor 1024 geheugenplaatsen van een byte groot. Een byte bestaat uit 8 bits (BInairy digiTs). Elke bit is een plaatsje in de computer dat aan of uit staat, een plus of een min, een een of een nul aangeeft, kortom een soort hele kleine schakelaar. Met deze enorme hoeveelheden 'schakelaars' kan de computer allerlei opdrachten voor u vervullen.
De ROM (Read Only Memory) is dat gedeelte dat door de gebruiker alleen gebruikt kan worden om gegevens uit te lezen. Nieuwe gegevens kunnen hier niet geplaatst worden. De ROM bevat in elke Atari-computer de BASIC-*interpreter* die 8 K byte geheugen in beslag neemt.
De BASIC interpreter is de tolk van de computer. De interpreter vormt de grammatika en het woordenboek, waarmee de computer de door de gebruiker in BASIC gegeven opdrachten kan begrijpen. Een computer werkt zelf binair. Denk aan de schakelaar (bits) die alleen een stand aan of uit, 1 of 0 kennen. De taal BASIC wordt alleen begrepen doordat de interpreter het vertaalwerk doet. Het is wel mogelijk de computer direkt in 1-en en 0-en aan te spreken, maar dit is voor de programmeur zeer lastig. Dit boek behandelt het programmeren in de taal BASIC.

De verschillen tussen de verschillende Atari-computers zijn:

Atari 400 - Een computer die niet meer in produktie is. De 600XL heeft deze computer vervangen. 16K aan geheugen en een plat toetsenbord. Bij gebruik van dit boek is het noodzakelijk de BASIC-cartridge in de computer te steken.

Atari 800 - Gelijk aan de 400, maar met een groter geheugen (48 K). Ook bij gebruik van deze machine met dit boek dient u de BASIC-cartridge aan te sluiten. Atari 600XL - De opvolger van de 400. Nu echter met een gewoon toetsenbord en ingebouwde BASIC.

Atari 800XL	- Gelijk aan de 600XL, maar met 48 K geheugen in plaats van 16 K.
Atari 65XE	- De nieuwste lijn computers van Atari. Opvolger van de 800XL. 48 K geheugen, modern uiterlijk en vriendelijk van prijs.
Atari 130XE	- De grote broer van de 65XE. Een gigantische hoeveelheid geheugen (128 K bytes). Een deel van dit geheugen kan dienst doen als extra 'diskdrive', of gebruikt worden als reservegeheugen bij het programmeren in machinetaal.

I/O en video

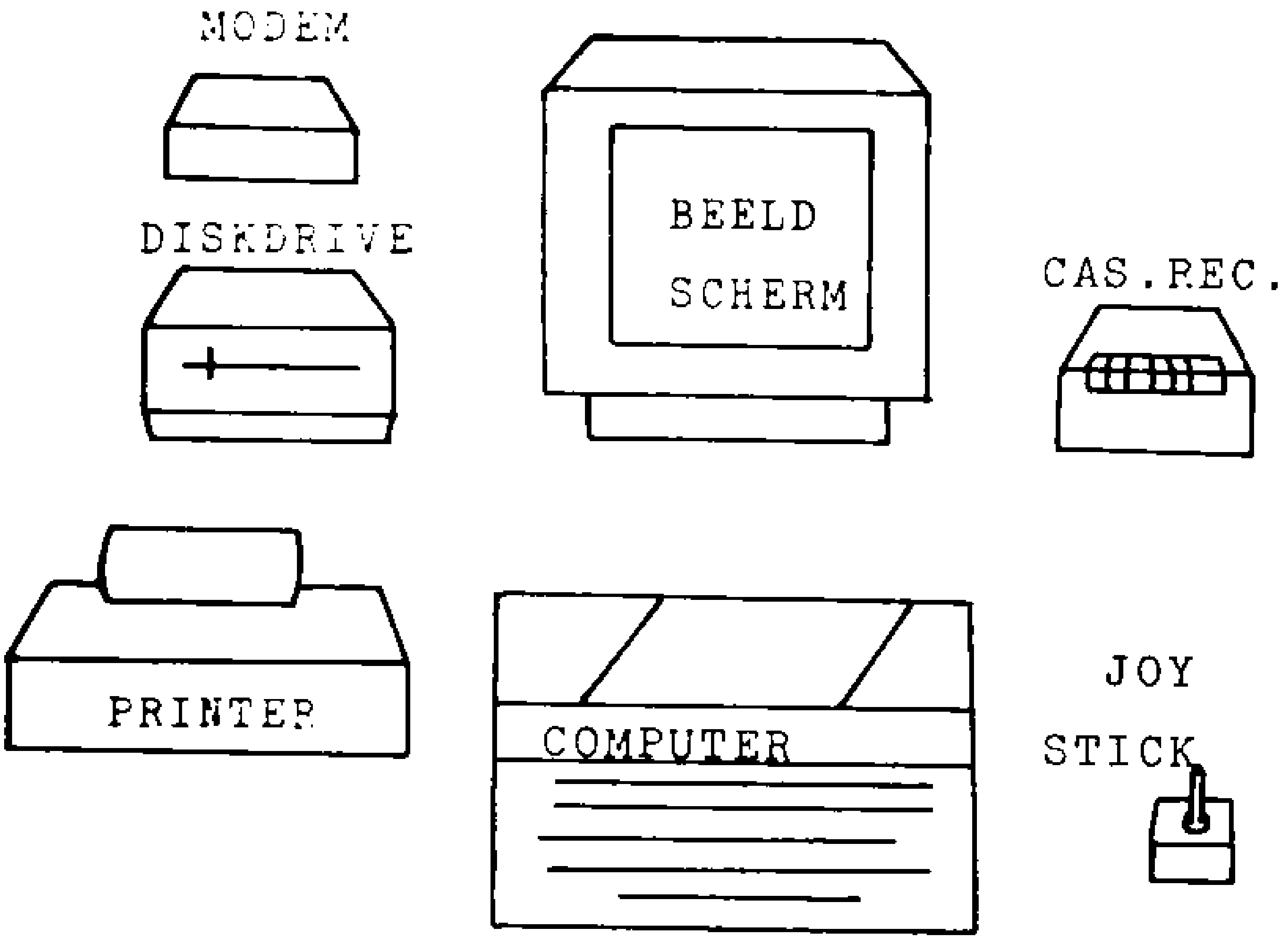
Over de twee andere delen van het schema valt iets minder te vertellen. De VIDEO zorgt voor de weergave op het beeldscherm. De Atari kan weergeven op een monitor (kleur of zwart/wit) of op één gewone televisie (kleur of zwart/wit). Het geluid dat de Atari kan voortbrengen wordt weergegeven door luidspreker van de televisie. Als u een monitor gebruikt, moet u een type nemen dat tevens geluid kan voortbrengen. Een monitor geeft een scherper beeld maar is duurder. Een televisie staat meestal al in huis, maar gezamenlijk gebruik zorgt (soms) voor ongemak door de keuze tussen het geweldige tv-programma XXX en een fascinerend computerprogramma. De I/O (Input/Output = invoer/uitvoer) zorgt voor de communicatie met randapparatuur zoals een printer, cassetterecorder, diskdrive etc.

De aansluiting

Atari-computers zijn niet moeilijk aan te sluiten. U zet de computer in de buurt van een televisie of een monitor. U sluit de antennekabel die met de computer is meegeleverd aan. De antennekabel van de televisie moet daarvoor uit de televisie-ingang worden gehaald. Heeft u een schakeldoosje bij de Atari gekregen dan gaan zowel de televisie-antennekabel als de computerkabel in de ingangen van dit doosje, en gaat het kabeltje van dit doosje in de antenne-ingang van de televisie. Nu kunt u, door de schakelaar om te halen, kiezen tussen een signaal van de antenne of een signaal van de computer voor uw televisie.
De kleine stekker van de transformator (die 220 volt omzet in een lager spanning die de computer gebruikt, dus nooit zelf aan gaan zitten rommelen!) steekt u in de POWER ingang van de computer. De grote twee-polige stekker gaat in een wandkontaktdoos (stopkontakt) van het lichtnet. Door de POWER-schakelaar om te halen, werkt uw computer. Vergeet niet dat de televisie aan moet staan, en afgestemd moet zijn op het signaal van de computer. Kies een weinig gebruikt kanaal en gebruik de fijnafstemming om het signaal van de computer zo goed mogelijk te ontvangen.
U kunt nu aan de slag met uw computer en met dit boek. Wilt u echter de door u gemaakte computerprogramma's bewaren, dan moeten die ergens buiten de computer opgeslagen worden. Als u de Atari uitzet, worden alle gegevens in

het geheugen gewist. Voor een definitieve opslag gebruikt u magnetisch materiaal. Op dezelfde wijze kan muziek op magnetisch materiaal (banden en cassettes) opgeslagen worden.
Er zijn twee vormen van magnetische opslag: cassettes en diskettes. Cassettes zijn goedkoop in gebruik. Om gegevens op cassettes op te kunnen slaan en later weer van de cassette te lezen heeft u een programma-recorder nodig. Bijvoorbeeld de Atari 1010.
Diskettes zijn veel sneller in gebruik dan cassettes, maar zowel de recorder die u ervoor nodig heeft (een diskdrive, de Atari 1050) als de diskettes zelf zijn duurder. Zolang u alleen op hobby-niveau werkt met de Atari en snelheid niet uw eerste vereiste is, bent u het beste uit met een programmarecorder met cassettes.

Naast de opslagapparaten kunt u nog verschillende andere zogenaamde 'randapparatuur' op uw Atari aansluiten.



Een printer biedt u de mogelijkheid om de resultaten van uw werk met de computer op papier vast te leggen. Dat is noodzakelijk als u uw Atari wilt gebruiken als schrijfmachine, maar ook voor vele andere werkjes is het lezen van papier minder vermoeiend dan lezen van het beeldscherm. Atari heeft een hele serie printers (in het nederlands ook wel 'afdrukeenheid' genoemd) in verschillende kwaliteits- en prijsklassen. U kunt kiezen uit printers die letters weergeven als een verzameling puntjes (dot-matrix printers) en printers die gesloten letters als van een schrijfmachine weergeven (daisy-wheel of vlinder-wiel printers).

Een van de meest gebruikte randapparaten van de Atari is de joystick of stuurknuppel. Met zo'n stuurknuppel kunt u de plaats van bijvoorbeeld een

ruimteschip op het scherm veel sneller en nauwkeuriger bepalen. Hoewel stuurknuppels vooral gebruikt worden voor spel-toepassingen kunnen zij ook dienen voor besturingsdoeleinden in serieuze toepassingen als tekstverwerking of CAD/CAM. Naast de stuurknuppels bestaan tegenwoordig ook **track-balls**. Een uitvinding die oorspronkelijk is gebruikt in de besturing van militaire tanks. Door met de vlakke hand over een bal te rollen kan het beeldscherm nog nauwkeuriger worden bestuurd.

Twee minder gebruikte randapparaten zijn het **touch-tablet** (aanraakvlak) en de **lichtpen.** Met het aanraakvlak kunnen twee-dimensionale tekeningen snel in de computer ingevoerd worden. De lichtpen biedt de gebruiker de mogelijkheid om direkt op het beeldscherm tekeningen te maken.

Een **modem** zorgt ervoor dat het signaal dat de computer afgeeft, wordt veranderd in een signaal dat over een telefoonlijn gestuurd kan worden. Andersom zorgt een modem er tevens voor dat een signaal dat over een telefoonlijn binnen komt, vertaald wordt in een signaal dat de Atari kan begrijpen. Op deze wijze is het mogelijk twee Atari's die ver van elkaar staan door een telefoonlijn met elkaar te laten converseren. Daarnaast is het met een modem mogelijk gegevens te halen uit grote gegevensbanken (bijvoorbeeld viditel).

De bediening

Atari-computers zijn niet veel ingewikkelder in het gebruik dan een electronische schrijfmachine. Helaas wordt bij verschillende types slechts een zeer summiere handleiding meegeleverd. Voor de volledigheid vindt u hieronder een korte inleiding in de bediening van de Atari-computers. Niet elk onderdeel is van toepassing voor elk type.

Testen

Alle nieuwere types Atari-computers kennen een zelf-test programma. Dit programma zit in de computer ingebakken en geeft de gebruiker de mogelijkheid zijn of haar computer op gezette tijden te controleren. Vooral het onderdeel dat het toetsenbord test, dient eens in de paar maanden te worden gebruikt. De toetsen van elk toetsenbord zijn aan slijtage onderhevig, en het is belangrijk dat eventuele defecten op tijd worden ontdekt.
U kunt het test-programma op twee verschillende manieren laten beginnen:
- Hou de OPTION toets ingedrukt bij het aanzetten van de computer en wacht tot het zelf-test-menu verschijnt.
- Als de computer al in BASIC staat, tikt u BYE en drukt u op de RETURN-toets.

De volgende tekst zal op het beeldscherm verschijnen.

```
SELF TEST
-------------------------
MEMORY
AUDIO-VISUAL
KEYBOARD
ALL TESTS
-------------------------
[illegible], [illegible] OR [illegible]
```

De tekst nodigt u uit, te kiezen uit een van de vier aangegeven mogelijkheden. De bovenste tekst MEMORY (=geheugen) staat te knipperen. Door de SELECT (=keuze) toets een of enkele malen in te drukken, kunt u zorgen dat een van de andere mogelijkheden gaat knipperen. Knippert die mogelijkheid die u wilt benutten, dan drukt u op START. De computer zal dan het door u gekozen test-programma afwerken. Wilt u alle testen doorlopen dan laat u met behulp van de SELECT toets de onderste tekst: ALL TESTS (=alle testen) knipperen en drukt u op de START toets. Als u een diskdrive heeft, dient u ervoor te zorgen dat deze uit staat tijdens het aflopen van het test-programma.

Memory (=geheugen)

Eerder in dit hoofdstuk heeft u al kunnen lezen dat de Atari twee soorten geheugen kent; het ROM en het RAM. In deze test worden ze allebei getest.

```
MEMORY TEST
ROM
█████████  █████████
RAM
▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮

[illegible] OR [illegible] TO EXIT
```

Het ROM gedeelte wordt aangegeven door twee balken die tijdens het testen wit zijn en na het testen groen of blauw worden. Worden de balken echter rood of paars dan is er iets niet in orde met het ROM geheugen en moet u met de computer naar de dealer.
Het RAM gedeelte wordt weergegeven door 48 witte blokjes die ook groen of rood kunnen worden. Elk blokje geeft een kilobyte geheugen weer. Ook de 130XE geeft 48 blokjes weer! Het extra geheugen van deze computer wordt met dit programma niet uitgetest.
Heeft u het test-programma verkregen door het tikken van BYE nadat u al in BASIC was beland, dan krijgt u maar 40 blokjes te zien. BASIC neemt immers 8 K bytes aan geheugen in beslag. Hoewel de BASIC zich ook in ROM

bevindt, wordt deze na het aanzetten in RAM gekopieerd. Daardoor kan de gebruiker naar wens de BASIC aanpassen.

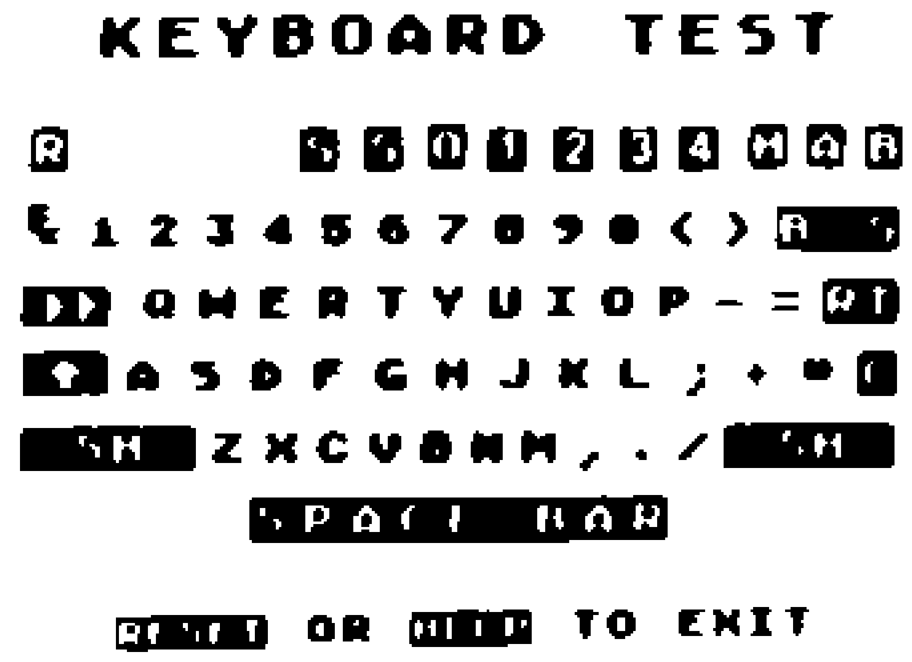
Audio-visual (geluid en beeld)

De Atari heeft uitstekende geluids- en grafiekmogelijkheden. Deze worden met dit programma uitgeprobeerd.

U ziet telkens een G-sleutel verschijnen met de weergave van enkele noten. Terwijl de noten op het scherm verschijnen, moeten deze ook hoorbaar zijn. Als u niets hoort moet u eerst controleren of het geluidsniveau van de televisie wel groot genoeg is. Een ander probleem kan zijn, dat de zenderafstelling wel goed is voor beeld, maar niet voor het geluid. Probeer of draaien aan de fijnafstemming helpt. Als u een monitor gebruikt dient u te kijken of deze geluid weer kan geven en of het geluidskanaal wel goed is aangesloten.
U zult vier verschillende stemmen (VOICE) horen. Daarna herhaalt het programma zich. Om te stoppen drukt u op de RESET (=herstel) of HELP toets.

Keyboard (toetsenbord)

Dit programma test de werking van het toetsenbord.

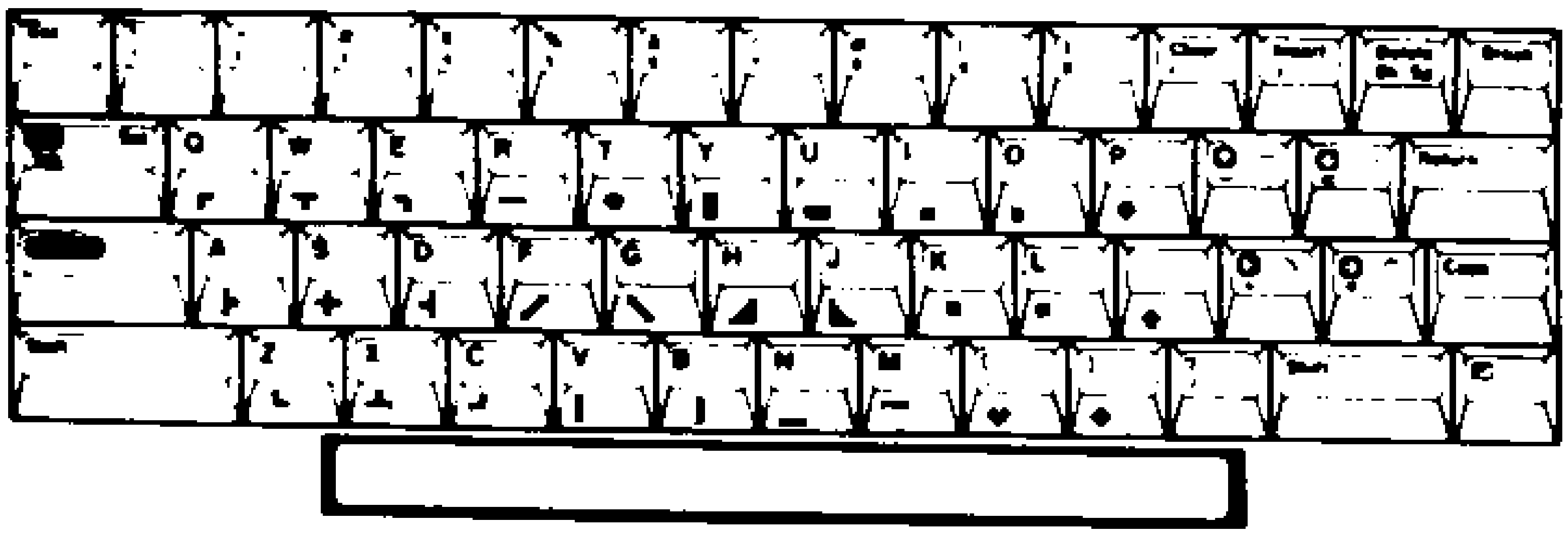


Alle toetsen van het toetsenbord zijn op het scherm weergegeven. Door telkens een toets in te drukken wordt dezelfde toets op het beeldscherm in omgekeerde kleur (inverse) weergegeven, terwijl een toeter klinkt. De SHIFT- en CONTROL toetsen geven alleen een resultaat als ze gelijk met een andere toets worden ingedrukt. De funktietoetsen zijn op een aparte wijze weergegeven.

BREAK (=onderbreek) geeft geen enkele reactie. HELP stuurt u terug naar het zelf-test programma. RESET stuurt u terug naar BASIC.

Het toetsenbord

Het toetsenbord van de Atari bevat alle toetsen die op een normale schrijfmachine voorkomen, maar heeft daarnaast ook een stel bijzondere toetsen. Omdat de communicatie met de computer meestal via het toetsenbord verloopt, is het nodig dat de programmeur een grondige kennis van de werking van het toetsenbord heeft. Oefent u vooral flink in het werken met het toetsenbord. Wat u ook intikt en welke toetsen u ook indrukt: u kunt uw computer **niet** beschadigen. Het ergste dat er kan gebeuren is dat u een programma verspeelt. Door de computer uit te zetten en een paar seconden later weer aan te zetten, heeft u de computer altijd weer terug gebracht in de oorspronkelijke staat.
De toetsen van het toetsenbord zijn gerangschikt volgens het QWERTY-patroon. Een patroon waarvan men een tijd geleden, toen de schrijfmachines ontstonden, dacht dat het snel tikte. Tegenwoordig zijn nog steeds bijna alle schrijfmachines met dit toetsenpatroon uitgerust. Bijna iedereen heeft er wel eens mee gewerkt, waardoor het voor de meeste gebruikers zeer vertrouwd is.



Hoewel de toetsenborden van de verschillende Atari-computers wel een beetje verschillen, treft u op elk toetsenbord de volgende onderdelen aan:

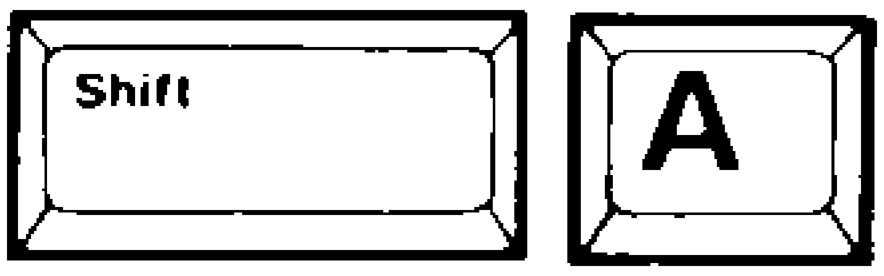
De lettertoetsen

Deze toetsen geven alle standaard gebruikte letters, cijfers en leestekens weer. Daarnaast kunnen met deze toetsen in samenhang met de andere toetsen verschillende tekens en symbolen worden weergegeven. Zie hieronder.

De speciale toetsen

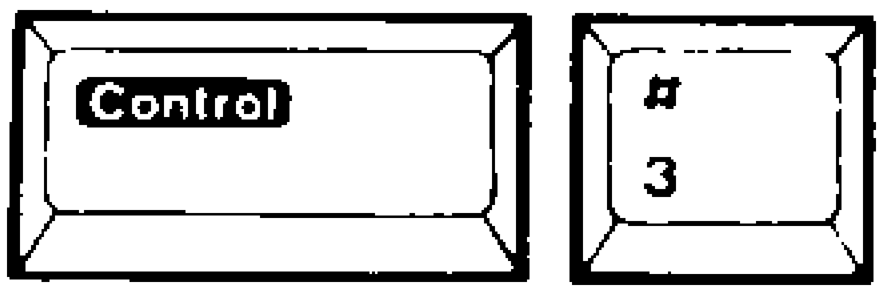
Zowel links als rechts op het toetsenbord vindt u enkele toetsen die bepalen hoe de tekens die met de normale lettertoetsen worden ingetikt, op het scherm verschijnen.
De SHIFT toets zorgt voor het weergeven van hoofdletters. Bij het aanzetten van de computer geeft deze alleen hoofdletters weer. Dat komt doordat de

hoofdletters zijn vastgezet. U ontsluit deze door de CAPS (van CAPitalS locked = hoofdletters gesloten) toets in te drukken. Als u daarna op de gewone lettertoetsen drukt verschijnen er kleine letters op het scherm. De SHIFT toets samen met een lettertoets ingedrukt, zorgt voor hoofdletterweergave.



SHIFT + letter

Door opnieuw de CAPS toets of de CAPS toets *tegelijk* met de SHIFT toets in te drukken, worden de hoofdletters weer vast gezet. Nu krijgt u weer alleen hoofdletters.

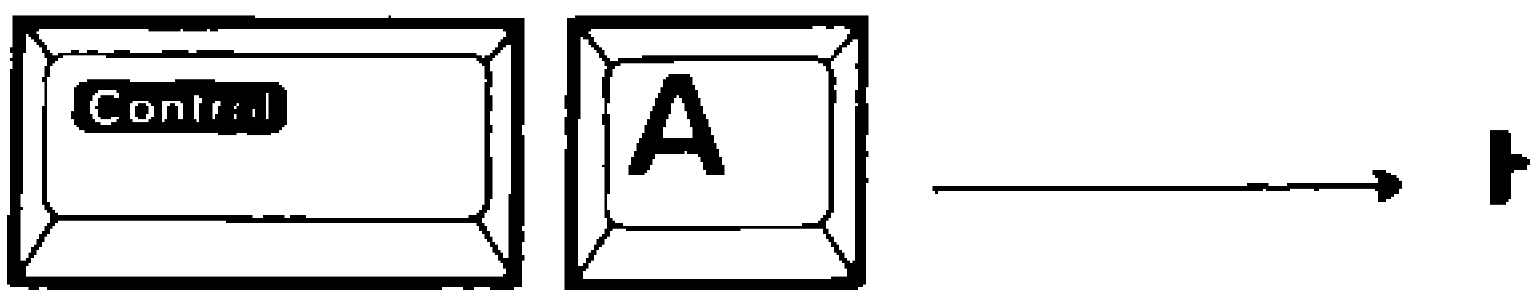


SHIFT + symbool

In de hoofdletters-gesloten stand geeft een symbooltoets het onderste symbool weer. Door deze toets *samen* met de SHIFT in te drukken krijgt u het bovenste symbool.

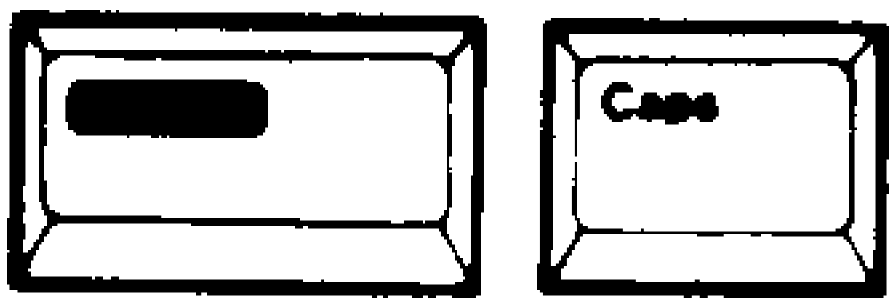
De CONTROL toets

Deze toets, de besturingstoets, zorgt voor speciale effecten als hij *tegelijk* met een lettertoets ingedrukt wordt. U krijgt dan de grafische symbolen die voor op de toetsen van de Atari 130XE zijn aangegeven. Eigenaren van andere Atari-computers kunnen in de bijlagen van dit boek opzoeken welke toets welk resultaat geeft.



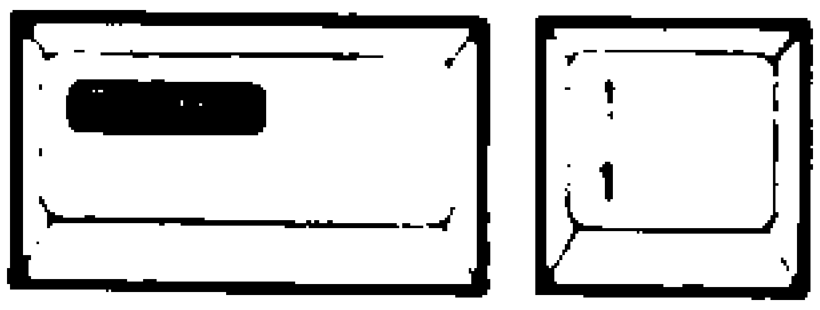
CONTROL + lettertoets = grafische symbool

Door CONTROL *samen* met CAPS in te drukken zet u het hierboven vermelde effect vast.



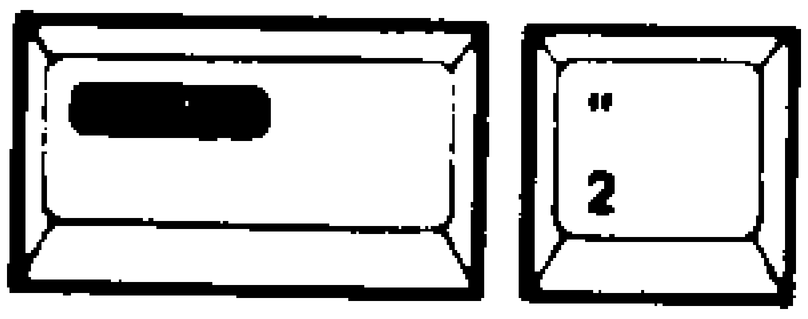
CONTROL + CAPS

Door de CAPS toets *nogmaals* in te drukken keren de lettertoetsen weer terug naar normale weergave.
Naast deze algemene funktie heeft de CONTROL toets nog een heel stel bijzondere funkties.



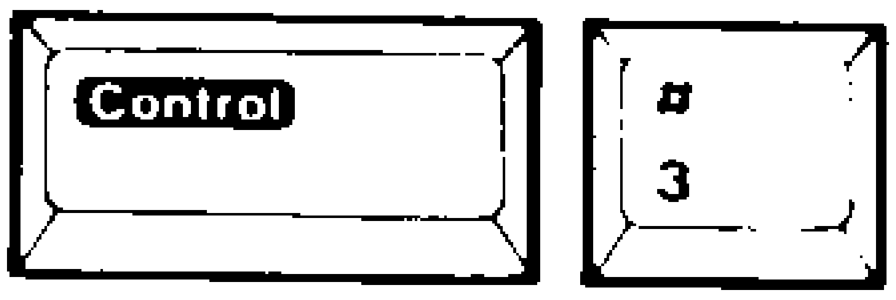
CONTROL + 1

CONTROL *tegelijk* met toets 1 zorgt voor het stoppen of weer starten van de weergave van een listing. Een listing is een lange lijst waarin een volledig computerprogramma staat. Soms zijn die programma's langer dan de hoogte van het beeldscherm. De bovenste regels verdwijnen dan boven uit beeld en onderin het beeld komen de nieuwe regels te voorschijn. Vergelijkbaar met de aftiteling van een speelfilm. De computer doet dit echter zo vreselijk snel dat u de tijd niet heeft om alles rustig te lezen. Door CONTROL en 1 tegelijk in te drukken stopt het doorrollen van het scherm. Dit doorrollen wordt ook wel, naar het Engels, scrollen genoemd. Nogmaals deze twee toetsen indrukken zorgt ervoor dat de listing verder gaat.



CONTROL+ 2

CONTROL *samen* met toets 2 zorgt voor het toeteren van de computer. Vergeet niet dat het geluid van de televisie (monitor) aan moet staan om het geluid te kunnen horen. Niet elke Atari heeft een eigen luidspreker.



CONTROL + 3

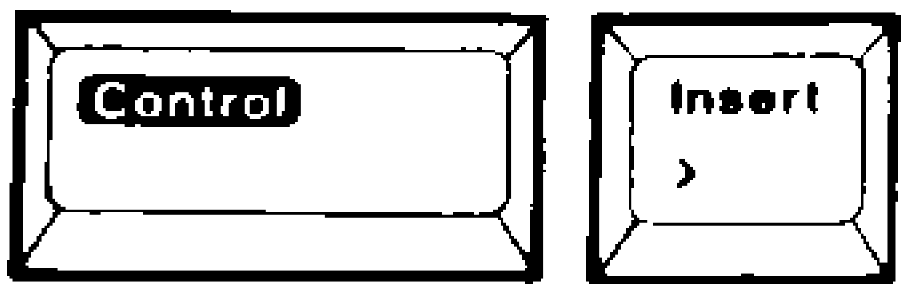
CONTROL *tegelijk* met toets 3 zorgt ervoor dat de computer een signaal krijgt dat het einde van het bestand is bereikt (end-of-file, EOF). Deze toepassing zult u alleen nodig hebben als u al een flink eind gevorderd bent met het leren programmeren. Tijdelijk kunt u dit ergens achterin uw geheugen opslaan.

Opmaaktoetsen

De Atari kent een heel stel toetsen die u helpen bij het 'opmaken' van het beeldscherm en het verbeteren van de door u gemaakte tikfouten.

Insert

De INSERT (=voeg in) toets zorgt voor het invoegen van spaties of witte regels.

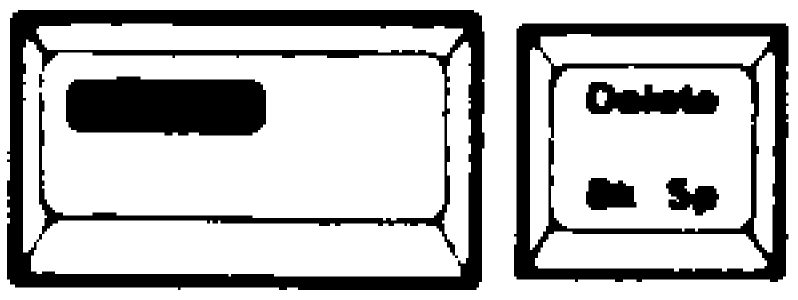


CONTROL+Insert SHIFT+Insert

CONTROL met INSERT *samen* zorgen voor een spatie op de plaats van de cursor. De cursor is de plaatsbepaler van de computer. U ziet deze als een wit blokje op uw scherm. Het geeft aan waar u met intikken bent gebleven. SHIFT *samen* met INSERT zorgt voor het tussenvoegen van een hele, lege regel.

DELETE

De DELETE (=verwijder) toets zorgt voor het verwijderen van een letterteken of van een hele regel lettertekens.

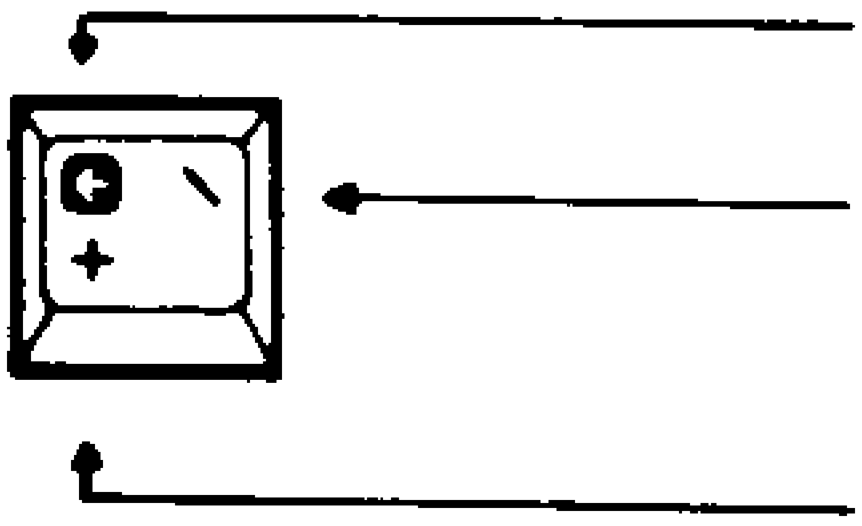


CONTROL+Delete SHIFT+Delete

Alleen de DELETE toets zorgt ervoor dat de cursor een plaats naar links schuift en het daar aanwezige letterteken weghaalt. De plaats die de cursor verlaat blijft leeg. De lettertekens rechts van de cursor worden niet verplaatst. CONTROL *met* DELETE verwijdert het letterteken onder de cursor. De overblijvende karakters rechts van de cursor worden allemaal een plaats naar links geschoven, zodat de tekst weer aansluit.
SHIFT *met* DELETE verwijdert een gehele regel van het scherm. Let op, als u dit doet bij het veranderen van een programmalisting. Door deze twee toetsen te gebruiken verwijdert u de regel wel uit de listing op het scherm, maar niet uit de listing in het geheugen van de computer. SHIFT en DELETE dienen *alleen* voor de beeldopmaak, niet voor het aanpassen van het programma.

Pijlen

Aan de rechterkant van het toetsenbord vindt u vier toetsen met pijltjes erop. Elke toets heeft drie symbolen. Elk van deze symbolen kunt u krijgen door de toets al dan niet samen met een andere toets in te drukken:



CONTROL + toets

SHIFT + toets

Alleen toets

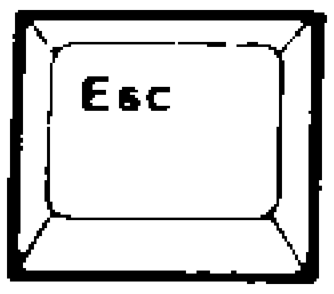
De verschillende funkties wijzen voor zichzelf. Behalve de funktie van de pijltjes. Met deze pijltjes kunt u de cursor willekeurig over het scherm verplaatsen. Hiermee kunt u verbeteringen aanbrengen in de door u ingetikte programmaregels. Het werkt extra snel, als u aan de onderkant van het scherm gekomen niet de pijl omhoog een hele tijd indrukt om bovenaan het scherm te komen, maar de pijl naar beneden nogmaals indrukt. De cursor verdwijnt dan

onderaan het scherm en verschijnt weer bovenaan. Andersom lukt ook, evenals links-rechts en rechts-links.

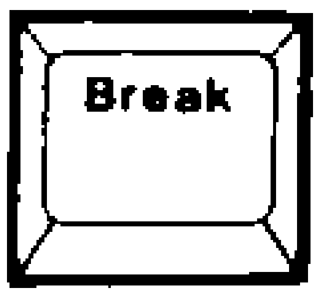
Speciale toetsen

Nog enkele speciale toetsen op het normale toetsenbord resten.

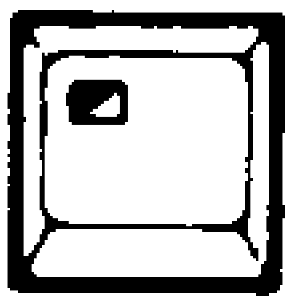
De RETURN toets is wellicht de belangrijkste en in ieder geval de meest gebruikte toets van uw computer. Telkens als u klaar bent met het geven van een opdracht of instruktie aan de computer drukt u op de RETURN toets. U moet dit vergelijken met het woordje 'over' bij het praten door een walkie-talkie. U geeft de computer een teken dat u klaar bent met intikken. Het woord is nu aan de computer.



De werking van deze toets is afhankelijk van het gebruikte programma. In Atari-BASIC werkt de ESCAPE toets als einde van de direkte werking van de cursor besturingstoetsen (o.a. de pijltjes). Door in een programmaregel eerst een keer ESCAPE in te drukken en vervolgens een cursorbesturingsopdracht binnen een PRINT-opdracht in te tikken (zie later in dit boek), wordt deze opdracht bij het uitvoeren van het programma verwerkt. U kunt dit zien door achtereenvolgens op ESCAPE en SHIFT DELETE te drukken, of op ESCAPE en een van de andere hierboven genoemde cursor besturingstoetsen.



Deze toets onderbreekt de werking van de computer. De computer houdt op met de taak waar hij mee bezig was, en geeft het woord weer aan de gebruiker.



Deze toets wordt op sommige Atari-computers aangegeven met het Atari-logo. Door het indrukken van de toets worden alle tekens in *inverse* weergegeven. Dat wil zeggen dat de voorgrond- en achtergrondkleur worden omgewisseld.

Funktietoetsen

Naast alle op het gewone toetsenbord voorkomende toetsen kennen de Atari-computers nog een vijftal aparte toetsen. Bij de XL-serie zijn die in chroom uitgevoerd en vindt u ze rechts van het toetsenbord. Bij de nieuwe XE-serie hebben de toetsen een parallelvorm en vindt u ze boven het toetsenbord.

HELP

Zorgt bij sommige programma's voor het weergeven van nadere instrukties voor de gebruiker die er niet meer uitkomt. De werking kan echter per programma verschillen.

START

Zorgt voor het beginnen van een bepaald programma. Zie bijvoorbeeld de zelf-test-programma's. Ook van deze toets kan de werking per programma verschillen.

SELECT

Zorgt voor het kiezen uit een serie mogelijkheden. De werking is afhankelijk van het gebruikte programma.

OPTION

Door deze toets ingedrukt te houden tijdens het aanzetten van de computer zal het zelf-test programma worden doorlopen. In een programma kan de werking van deze toets worden veranderd.

RESET

Deze toets stopt de computer met zijn bezigheden. U krijgt het scherm te zien zoals dat is na het aanzetten van de computer. Door op RESET te drukken krijgt u hetzelfde resultaat als het uit en aan zetten van de computer, met als verschillen dat het RAM geheugen niet gewist wordt (waardoor het program-

ma niet verloren gaat) en dat de aan en uitknop en de transformator minder slijten.

Internationale lettertekens

De Atari 130XE kent een *extra* stel internationale karakters. U krijgt deze door de volgende opdracht in te tikken, gevolgd door RETURN.

POKE 756,204

Als u nu de CONTROL-toets *samen* met de lettertoetsen gebruikt krijgt u geen grafische tekens maar lettertekens met accenten. Om weer terug te keren naar normale letters en grafische symbolen tikt u de opdracht:

POKE 756,224

gevolgd door RETURN.

Schermopmaak 2

Naast het gebruik van de toetsen zijn de volgende faciliteiten van de Atari voor de gebruiker van belang.

Auto repeat

Auto repeat wil zeggen dat de werking van alle toetsen van de Atari zichzelf herhalen als u de toets wat langer ingedrukt houdt. Probeert u dit maar uit door een een willekeurige toets ingedrukt te houden.

Scherm wissen

U kunt het gehele beeldscherm van tekst wissen door de CONTROL-toets *gelijk* met de CLEAR-toets in te drukken. Hetzelfde effekt bereikt u met SHIFT en CLEAR

Tabuleren

De TAB-toets links op het toetsenbord zorgt ervoor dat de cursor naar de volgende TAB-positie op het scherm springt. U kunt zelf nieuwe TAB posities maken door de cursor op de gewenste positie te plaatsen (gebruik de pijltoetsen), en SHIFT met TAB in te drukken. U heft een TAB-positie op door de cursor op de te verwijderen TAB positie te plaatsen en CONTROL TAB in te drukken.

Gebruik van een diskdrive

Bij het aanzetten van de computer dient u enkele afwijkende handelingen te verrichten als u gebruik maakt van een diskdrive. Het programma dat de diskdrive bestuurt, bevindt zich op een speciale diskette. U moet dit programma eerst laden om later de diskdrive te kunnen gebruiken. Laat u dit na, dan loopt u de kans op een gegeven moment een programma geschreven te hebben, dat u vervolgens niet op een diskette kunt wegzetten omdat u het disk-operating-systeem (DOS) niet geladen heeft. Tot op heden leverde Atari DOS 2.0 of DOS 3.0 met een diskdrive mee. Om zo'n programma te laden steekt u eerst de master-diskette (de diskette met daarop het DOS programma) in de diskdrive, zet deze aan, wacht enkele ogenblikken en pas dan zet u de computer aan. Na enkele momenten verschijnt de melding

READY (klaar)

Om DOS in het geheugen van de computer te krijgen tikt u nu

DOS (gevolgd door RETURN)

U zult nu een zogenaamd menu zien. Een menu is een lijstje met mogelijkheden om uit te kiezen. De twee mogelijkheden die nu van belang zijn, zijn:

- File index
- To cartridge

File index kiest u door een F in te tikken. De computer vraagt nu om een Filespec? (bestandspecifikatie). U tikt nu:

. (gevolgd door RETURN)

Nu vraagt de computer om een display device (apparaat waarop weergegeven moet worden). U kunt volstaan met het tweemaal indrukken van de RETURN toets.
U krijgt nu een lijst te zien van alle programma's en bestanden op de diskette die nu in de diskdrive zit. Wilt u de lijst met programma's van een andere diskette zien, dan wisselt u de diskette in de diskdrive voordat u voor de laatste maal op RETURN drukt.
Om terug te gaan naar het menu drukt u weer op RETURN.

To cartridge kiest u door een T in te drukken. De computer schakelt nu over naar BASIC. Het lijkt vreemd dat, om naar BASIC te gaan, u kiezen moet voor het naar een cartridge gaan. Dit komt doordat de BASIC in de oudere Ataricomputers zich in een aparte BASIC-cartridge bevond.

Het werken met het *allernieuwste* DOS 2.5 is gelijk aan het werken met DOS 3.0 of DOS 2.0. Het menu ziet er nu echter iets anders uit. Ten eerste moet u niet meer de eerste letter van de opdracht intikken, maar de letter die los van

de opdracht ervoor is vermeld. Dat wil zeggen dat u voor BASIC kiest voor de letter **B** (RUN CARTRIDGE).
File index is vervangen door DISK DIRECTORY (inhoud van de diskette)

Als u met diskettes werkt dient u niet te vergeten dat de Atari alleen gebruik kan maken van diskettes die speciaal voor de Atari aangemaakt (heet formateren) zijn. Dat aanmaken doet u door na het laden van DOS te kiezen voor de mogelijkheid FORMAT DISK (I in DOS 3).
De computer vraagt u dan welke diskdrive geformateerd moet worden. Als u één diskdrive heeft tikt u nu een 1 gevolgd door RETURN. Haal nu de programmadiskette uit de diskdrive! **Het formateren van een diskette wist alle gegevens die zich daarop bevinden.**
Stop een nieuwe diskette in de diskdrive en druk op de Y-toets van yes (ja) gevolgd door RETURN. De diskdrive gaat nu een rommelend geluid produceren en maakt de diskette klaar voor gebruik met de Atari.

Zie hiervoor ook bijlage 6.

3. DIREKTE OPDRACHTEN

De eenvoudigste wijze om uw Atari te gebruiken is hem een opdracht te geven (in BASIC natuurlijk, want anders begrijpt hij u waarschijnlijk niet). Enkele direkte opdrachten die niet tot BASIC horen, heeft u hiervoor al kunnen lezen. Zo kunt u door de CONTROL en de 2-toets *tegelijk* in te drukken de computer opdracht geven om een bliep te laten horen. Deze opdracht wordt direkt uitgevoerd.
Om nu eens een BASIC-opdracht uit te proberen moet u de computer eerst weer in de begintoestand brengen. Druk op RESET of onderbreek de stroomtoevoer (aan/uit-knop, stekker) gedurende enkele seconden. Als u een diskdrive heeft, laadt u eerst DOS (daar moet u altijd mee beginnen) en gaat u vervolgens naar Atari-BASIC.
Tik nu:

```
PRINT "Ik laat een zinnetje zien."
```

en druk op de RETURN-toets.
U ziet, de computer voert onmiddellijk uit wat u hem opdraagt, namelijk het afdrukken (PRINT) van een korte tekst.
Heeft u door gebruik te maken van de SHIFT- en eventueel de CAPS-toets op precies dezelfde plaatsen als hierboven hoofdletters gebruikt? Goed voor de vingervlugheid! Denk eraan dat de tekst wel in kleine letters geschreven kan worden, maar dat de opdracht aan de computer (PRINT) in Atari-BASIC altijd in *hoofdletters* ingetikt moet worden. Probeert u de volgende opdracht maar eens:

```
print "Ik laat een zinnetje zien."
```

Door middel van de RETURN-toets geeft u de machine te kennen dat u klaar bent met het intikken van een regel met opdracht(en) en dat de computer deze moet uitvoeren (bij direkte opdrachten) of in zijn geheugen moet opslaan (bij indirekte opdrachten).
In dit geval begrijpt de Atari de opdracht niet en geeft een foutmelding (ERROR=vergissing). De cursor wordt op de eerste gevonden fout in een opdracht geplaatst. In dit geval de 'p' van 'print' omdat de computer alleen een opdracht in hoofdletters accepteert (PRINT).

De aanhalingstekens aan weerszijden van de tekst laten de computer weten dat de daartussen geplaatste tekst niet bedoeld is als BASIC-opdracht, maar als tekst die geheel moet worden afgedrukt. Tikt u ter controle maar:

```
PRINT Ik laat een zinnetje zien.
```

en druk op de RETURN-toets.
Nu begrijpt de computer toch niet helemaal wat de bedoeling was. De

opdracht wordt niet uitgevoerd en u krijgt een foutmelding met de cursor op de 'k' van 'Ik'. Probeert u uit wat er gebeurt als u de zin in hoofdletters opgeeft. U krijgt dan wederom een foutmelding met de cursor op de 'L' van 'LAAT'. Waarom staat in het ene geval de cursor bij de foutmelding op de 'k' en in het andere geval op de 'L'? In beide gevallen ziet de computer de door u opgegeven tekst als een variabele en niet als een stukje tekst. Wat een variabele is kunt u in een van de volgende hoofdstukken nalezen.
Net als bij opdrachten is de Atari tamelijk kritisch bij het accepteren van variabelenamen. Deze mogen alleen uit hoofdletters bestaan, en de letters moeten aan elkaar worden geschreven. Daarom staat in het eerste voorbeeld de cursor op de 'k' (geen hoofdletter) en bij het tweede voorbeeld de cursor op de 'L' (spatie tussen IK en LAAT). Controleert u dit door in te tikken:

```
PRINT IKLAATEENZINNETJEZIEN
```

De computer geeft nu als antwoord: 0. De tekst wordt gezien als variabele waarvan de waarde nog niet opgegeven is, en dus de waarde 0 heeft. Meer hierover leest u in het hoofdstuk 'Variabelen'.
Alleen door voor en achter de tekst een stel aanhalingstekens te zetten begrijpt de computer dat u een stuk tekst opgeeft waaraan niets mag worden veranderd. Zo'n tekstje, zo'n rij van lettertekens heet een **STRING** (=sliert). U zult dit nog vele malen in dit boek tegenkomen.

De Atari voert dus alleen goed opgegeven opdrachten uit. Vergelijkt u de volgende twee opdrachten maar:

```
SOUND 1,100,2,2
SOUUND 1,100,2,2
```

of:

```
SETCOLOR 4,15,6
STCOLOR 4,15,6
```

Bij de onderste van elk van deze opdrachten krijgt u een foutmelding te zien die onveranderlijk in de vorm van ERROR gegeven wordt. Om de monotone brom uit te schakelen tikt u ofwel

```
SOUND 1,0,0,0
```

ofwel u drukt op de RESET knop.

Fouten verbeteren

Bij het verbeteren van fouten zijn de pijltoetsen en de andere opmaaktoetsen bijzonder handig. Stel u heeft als laatste opdracht de foutieve opdracht

STCOLOR 4,10,6

ingetikt. De computer zal nu reageren met:

ERROR- STCOLOR 4,10,6

Om deze fout te verbeteren drukt u tweemaal *tegelijk* de CONTROL en de pijl- omhoog toets in. De cursor staat nu op de S. Nu drukt u eenmaal de CONTROL toets gelijk met de pijl naar rechts in. De cursor komt daarmee op de T te staan. Door CONTROL *samen* met INSERT in te drukken, blijft de cursor op zijn plaats staan, maar wordt de tekst rechts ervan een positie naar rechts geschoven. Daardoor komt een letterpositie vrij. Deze vult u door de ontbrekende letter (een E) in te tikken. U bent nu klaar met het verbeteren van de opdracht. Nu kunt u, door op RETURN te drukken, de computer opgeven dat u op dit moment niets meer mee te delen hebt. De computer voert de opdracht, die nu wel begrijpelijk is, onmiddellijk uit: de rand van het beeldscherm wordt lichtgroen.
De tekst met ERROR blijft staan, maar doör het veranderen van de kleur van de rand van het beeldscherm heeft u kunnen zien dat de computer in dit geval de opdracht begreep. Er verschijnt ook de mededeling

READY (=klaar)

waarmee de computer aangeeft dat hij klaar is met de opdracht en op een nieuwe opdracht wacht.

Probeer de volgende drie opdrachten ook uit:

PRINT "Ik laat een zinnetje zien.
PRINT Ik laat een zinnetje zien."
? "Ik laat een zinnetje zien."

Gebruik hierbij de opmaaktoetsen om het intikken te beperken.
Tik eerst het bovenste voorbeeld in. De computer reageert niet op uw vergissing (de aanhalingstekens aan het einde van de string zijn weggelaten). De zin wordt gewoon afgedrukt. Handig als u eens iets vergeet, maar in een programmaregel met meer dan een opdracht kan dit vervelende effecten teweeg brengen.
Tik de tweede zin nu niet weer helemaal, maar verplaats de cursor vier maal omhoog en zesmaal naar rechts. De cursor staat nu op het eerste stel aanhalingstekens. Druk tegelijk op CONTROL en DELETE. De aanhalingstekens verdwijnen. Verplaats de cursor naar het einde van de zin en plaats dan de aanhalingstekens. Druk op RETURN.
In het tweede voorbeeld ziet de computer nu weer een variabele in de tekst. Wen u er aan om een string van lettertekens altijd tussen aanhalingstekens aan beide zijden te zetten. Dat is de enige manier waarop de computer foutloos zal reageren.

Om het derde voorbeeld te krijgen verplaatst u de cursor drie plaatsen omhoog en verwijdert u met CONTROL en DELETE het woordje PRINT. Met CONTROL INSERT maakt u ruimte om het vraagteken in te tikken. Maak weer ruimte en tik de eerste aanhalingstekens. Als u nu op RETURN drukt geeft de computer het volgende cryptische beeld:

```
? "Ik laat een zinnetje zien."
Ik laat een zinnetje zien. zinnetje zi
en."
READY
READY
```

U begrijpt natuurlijk al wat er gebeurd is. De computer heeft de laatste opdracht geaccepteerd en keurig het zinnetje afgedrukt. Dit ging echter over de oude zin heen, die daardoor gedeeltelijk op het scherm blijft staan. Ook de READY van de eerste keer blijft staan, zodat het beeld als geheel nogal rommelig wordt.

Belangrijk is dat het derde voorbeeld laat zien dat in plaats van de opdracht PRINT ook het korte ? ingetikt kan worden. Ook na deze vorm van de PRINT-opdracht moeten aanhalingstekens worden gebruikt!

Vergeet u bij het intikken van opdrachten en programmaregels met opdrachten de volgende punten niet:

Alle toetsen zijn auto-repeat (=zelf herhalend). Dat wil zeggen dat als u een toets een tijdje ingedrukt houdt, deze zichzelf gaat herhalen. Dit geldt ook voor de pijltoetsen.

Als u het geluid van uw televisie of monitor aan heeft staan, hoort u dat bij elke toetsaanslag een tikje klinkt. U kunt hiermee op het gehoor controleren of u de toets goed ingedrukt heeft.

Elke (regel met een) opdracht moet afgesloten worden door op de RETURN-toets te drukken. Hierdoor wordt de opdracht de computer ingevoerd, en kan de computer aan de uitvoering beginnen, of de opdracht of de regel met opdrachten opslaan in het geheugen.

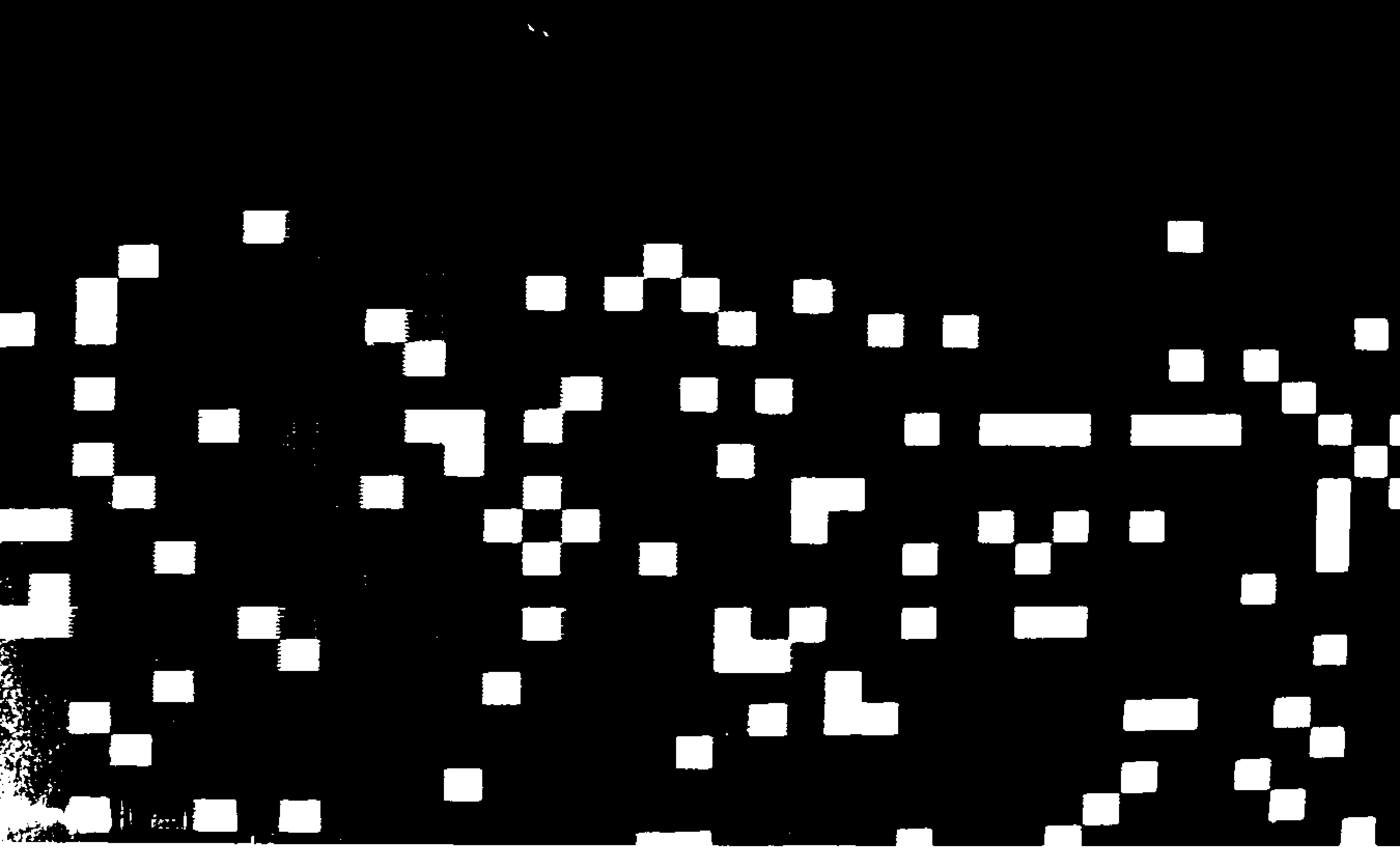
Vanaf hier wordt kortheidshalve het indrukken van de RETURN-toets niet aangegeven. U moet na elke opdracht, of na elke programmaregel deze toets indrukken!

Opdrachten

We proberen nog een opdracht:

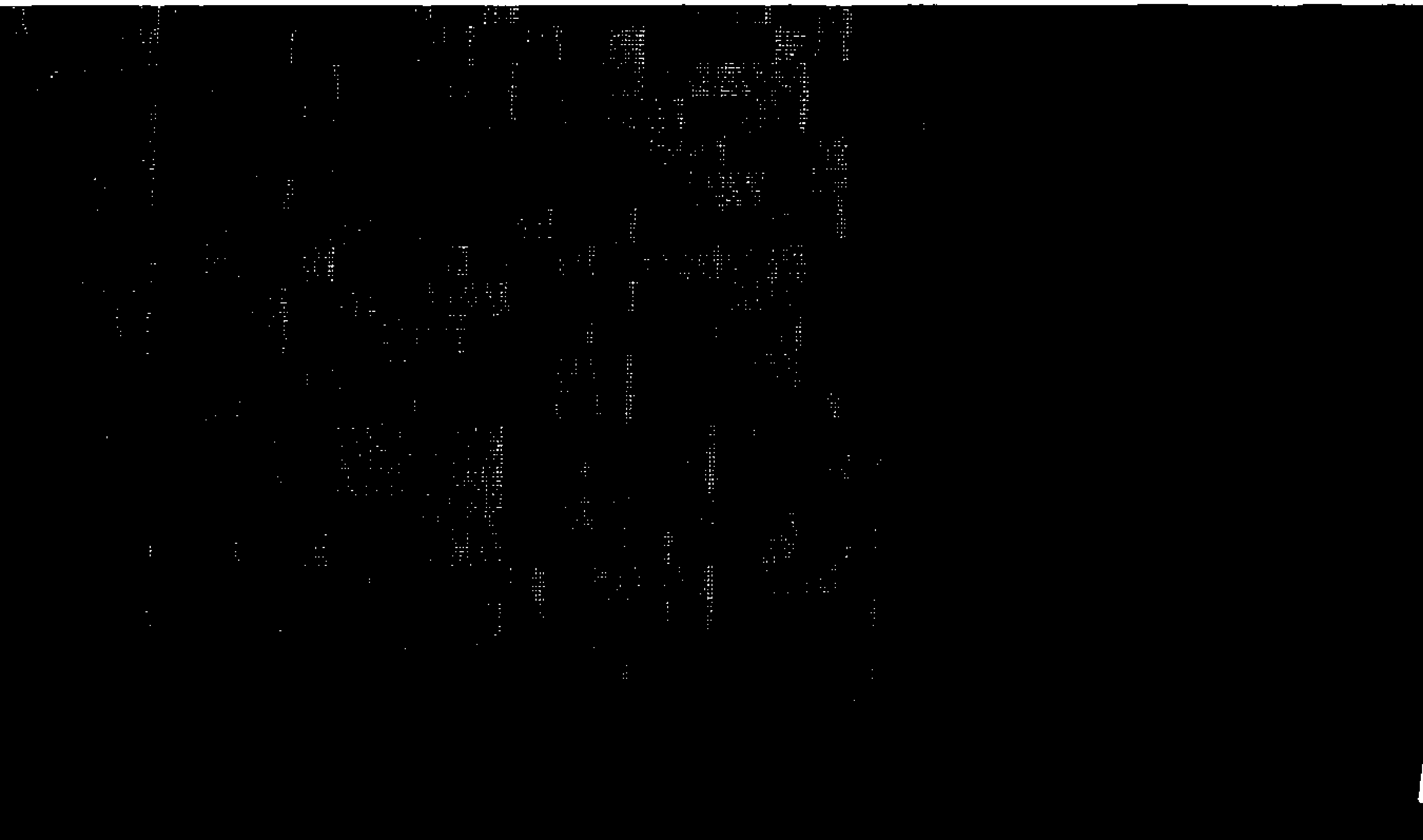
```
PRINT 13+6
```

Zonder dat u uw Atari ook maar iets heeft verteld kan hij al optellen. En niet alleen dat, hij kan alles wat uw rekenmachine ook kan. (Bewaart u die rekenmachine toch nog maar even. Die is immers iets compacter en handiger te vervoeren dan een Atari met televisie.) Probeert u maar:



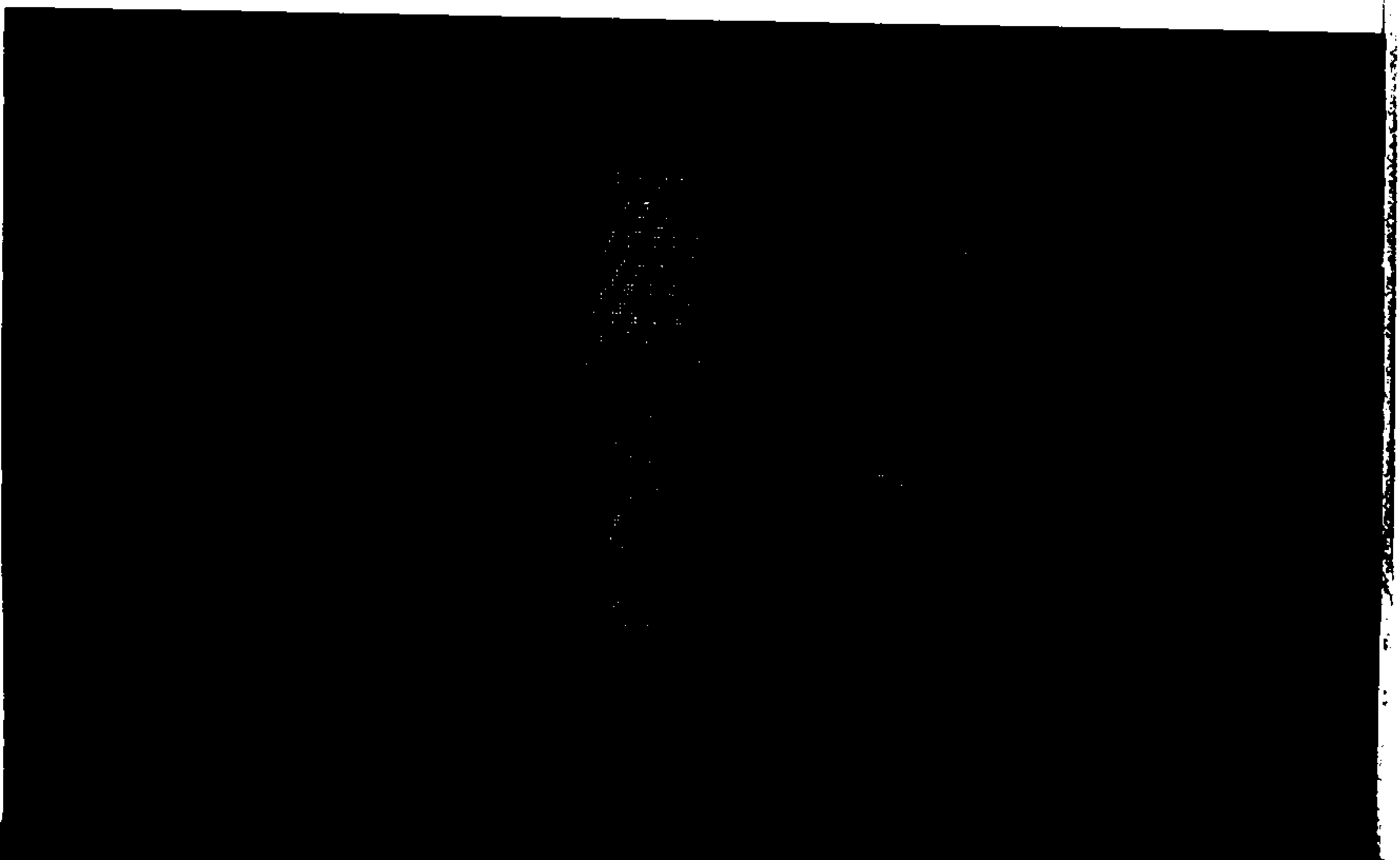
Gekleurde blokjes programma *blz. 188*

Graphics voor soortgelijke programma's *hoofdstuk 21*



Regenboog programma *blz. 155*

Kleurovaal programma *blz. 164*



```
PRINT 12-6
PRINT 12/6
PRINT 12*6
PRINT 12^6
```

U ziet: perfectie! Let u wel op de iets afwijkende notatie in BASIC. Het vermenigvuldigingsteken is een asterisk (*). Het deelteken is een schuine streep (/). Machtsverheffen noteert u niet door de exponent schuin boven het te verheffen getal te plaatsen, maar door een pijltje of dakje (circumflex) (^) gevolgd door de exponent te plaatsen.
Met negatieve getallen heeft de Atari natuurlijk ook geen problemen. Probeert u zelf 12 van 6 af te trekken. Ook lastiger berekeningen geven geen problemen:

```
PRINT 13*6+2*5^2/4+13^2*13
```

Krijgt uw Atari er ook tweeduizendtweehonderdzevenentachtig en een half uit?
De Atari houdt bij berekeningen keurig de volgorde aan die gebruikelijk is in de rekenkunde: **'Meneer Van Dalen Wacht Op Antwoord'**. Eerst Machtsverheffen (^), dan Vermenigvuldigen en Delen (* en /), dan Worteltrekken (SQR) en dan Optellen en Aftrekken (+ en -). Bij gelijkwaardige bewerkingen zoals optellen en aftrekken werkt de Atari van links naar rechts.
Als u een bepaalde bewerking voor een andere wilt laten uitvoeren, kunt u gebruik maken van **haakjes:**

```
PRINT 3*5^2
PRINT (3*5)^2
```

In het eerste geval wordt 5 gekwadrateerd (25) en daarna met 3 vermenigvuldigd (75). In het tweede geval wordt 5 met 3 vermenigvuldigd (15) en het resultaat gekwadrateerd (225).

In alle hierboven gegeven voorbeelden kunt u de PRINT opdracht vervangen door het veel kortere ?
Er bestaan naast de PRINT-opdracht nog verschillende andere direkte opdrachten. Bijna alle BASIC-opdrachten die de Atari kent, kunnen direkt worden gegeven. Denk bijvoorbeeld aan de hiervoor genoemde SOUND (=geluid) en SETCOLOR (=kleurinstelling) opdrachten. Veel van deze direkte opdrachten zult u later in het boek nog tegenkomen. Belangrijk is dat u gemerkt heeft dat de computer geen 'programma' nodig heeft om een opdracht te begrijpen. Het indrukken van RETURN na een opdracht is voldoende om de computer een opdracht uit te laten voeren. Hierdoor kunt u de computer ook voor allerlei snelle kleine karweitjes gebruiken. Deze kleine snelle direkte opdrachten zullen vaak van pas komen bij het bedenken, uitvoeren en schrijven van langere computerprogramma's.

Vindt u na alle voorbeelden uit dit hoofdstuk het beeldscherm wat rommelig dan drukt u *tegelijk* de CONTROL en de CLEAR toets in, of *tegelijk* de

SHIFT en CLEAR toets. U geeft de computer daarmee een direkte opdracht die zonder nadere aanduiding door middel van RETURN wordt uitgevoerd. U kunt ook een direkte opdracht met RETURN geven. Om het beeld te wissen gebruikt u:

GRAPHICS 0

Uiteraard gevolgd door RETURN. In dit geval voert de computer een echte direkte opdracht uit en geeft, door het afdrukken van de tekst READY, aan dat hij klaar is.

De laatste mogelijkheid om het beeldscherm te wissen is het indrukken van de RESET knop. **Wees daar echter voorzichtig mee, want u wist daar meer mee uit dan alleen het beeldscherm!**

4. EEN PROGRAMMA SCHRIJVEN

Een programma

Tikt u het volgende programma in. Na elke regel drukt u natuurlijk op RETURN.
Na de invoer van elke regel verschijnt links in beeld de cursor: u kunt de volgende regel intikken. Als u in een regel een fout heeft gemaakt geeft de Atari dat aan. Verbeter eerst de fout en ga dan verder met de volgende regel. Het hindert niet als er foutmeldingen tussen de regels van het programma in blijven staan. U moet er alleen voor zorgen dat de laatste versie van elke ingetikte regel zonder fouten is.

```
10 REM EEN VERMENIGVULDIGING
20 LET A=99
30 LET B=12
40 LET P=A*B
50 PRINT A;" * ";B;" = ";P
```

De cursor staat klaar voor de volgende regel. Die is er niet. U tikt:

RUN (+ RETURN)

Het blijkt dat de Atari ook verschillende opdrachten na elkaar kan uitvoeren. Dat had u natuurlijk al verwacht, want waar heb je anders een computer voor.

Let u bij het geven van opdrachten op de volgende punten:

- In een BASIC-programma begint elke regel met een positief geheel getal (het regelnummer).
- De computer voert de verschillende regels in volgorde van de regelnummers uit.
- Op een regel kunnen verschillende opdrachten staan (zie bijvoorbeeld regel 50: de Atari moet volgens die regel 5 verschillende zaken weergeven op het scherm. Dit is eigenlijk een opdracht met verschillende onderdelen. Later zult u echter zogenaamde multi-statement-regels (regel met meer dan een statement = opdracht, funktie of bewering) tegenkomen.
- Het is gebruikelijk bij het schrijven van een programma de regelnummers per 10 te laten verspringen. Op deze manier kunt u later nog op eenvoudige wijze regels tussenvoegen.
- U hoeft de regels niet in volgorde in te tikken. De Atari voert ze echter wel in de juiste volgorde uit, en zet ze in volgorde in het programma.

(De opdracht LET geeft aan een algemene waarde -een zogenaamde variabele-een bepaalde waarde. LET A=99 wil zeggen: vanaf nu is A gelijk aan 99. LET

P=A*B wil zeggen: vanaf nu is P gelijk aan A vermenigvuldigd met B. We komen in het hoofdstuk 'Variabelen' hier nog uitgebreid op terug.)

Hulp bij het maken van programma's

Er zijn verschillende hulpjes in de Atari ingebouwd om het programmeren gemakkelijker en sneller te maken. De mogelijkheid om fouten te verbeteren met behulp van de verschillende opmaaktoetsen heeft u in de vorige hoofdstukken al leren kennen.
Om het gehele ingetikte programma in een keer te kunnen zien tikt u:

LIST

gevolgd door RETURN. Is het programma langer dan een beeldscherm vol, dan verdwijnen de bovenste regels. Probeert u dit uit door een programma te maken van 28 regels, met op elke regel alleen een ?. Doordat de regels bovenaan het scherm weer zo snel verdwijnen is het erg lastig een lang programma te lezen. Om rustig steeds een deel van de programmalisting te lezen gebruikt u de CONTROL toets *samen* met de 1. Drukt u deze twee tegelijk in, dan stopt het doorrollen (scrollen) van de listing van het programma. Door beiden *nog* een keer in te drukken loopt de listing weer verder. Wilt u slechts een enkele regel uit het programma zien, dan tikt u:

LIST regelnummer

Voor 'regelnummer' tikt u het nummer van de regel die u wilt lezen. Om een paar regels te zien kunt u een van de volgende opdrachten tikken:

LIST beginregel , eindregel

Met deze opdracht kunt u elk deel van de listing bekijken. De regelnummers van de eerste te listen regel en van de laatste te listen regel hoeven niet in het programma te bestaan. Om de eerste 5 regels te zien kunt u bijvoorbeeld tikken:

LIST 0,50

Om alle regel vanaf 150 te zien kunt u tikken:

LIST 150,3000

Daarbij neemt u als laatste te listen regel, een regelnummer waarvan u zeker weet dat het groter is dan het laatste ingetikte regelnummer.

Een programma maken

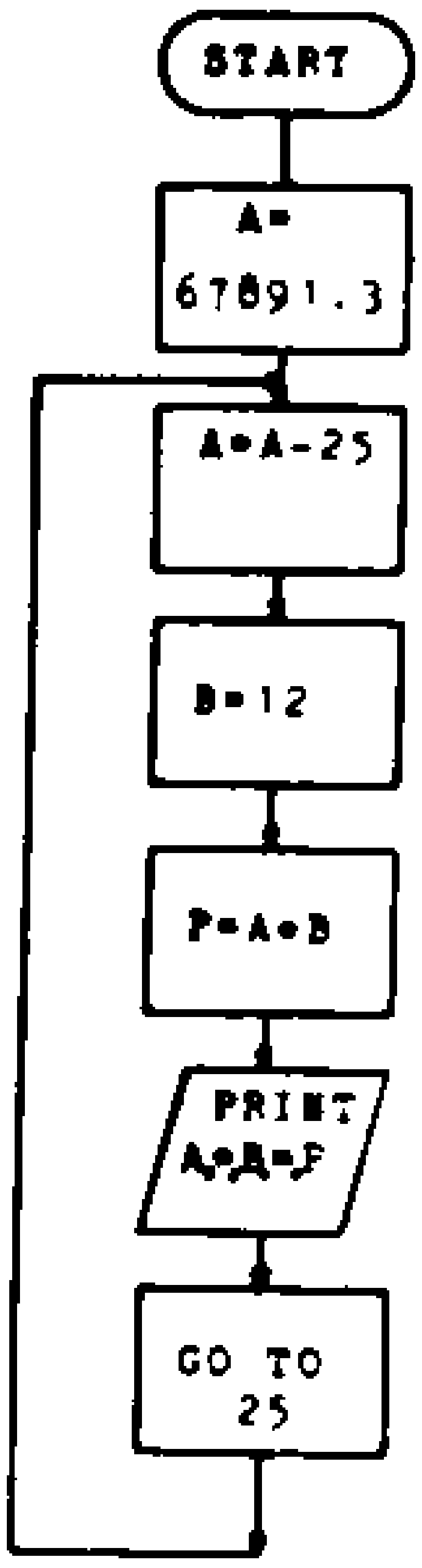
Nu we de mogelijkheid hebben om verschillende opdrachten na elkaar uit te laten voeren, waarbij we weten dat de computer zich houdt aan de regelnummers, hebben we de beschikking over een systeem met verschillende voordelen. Tik:

```
20 LET A=67891.3
```

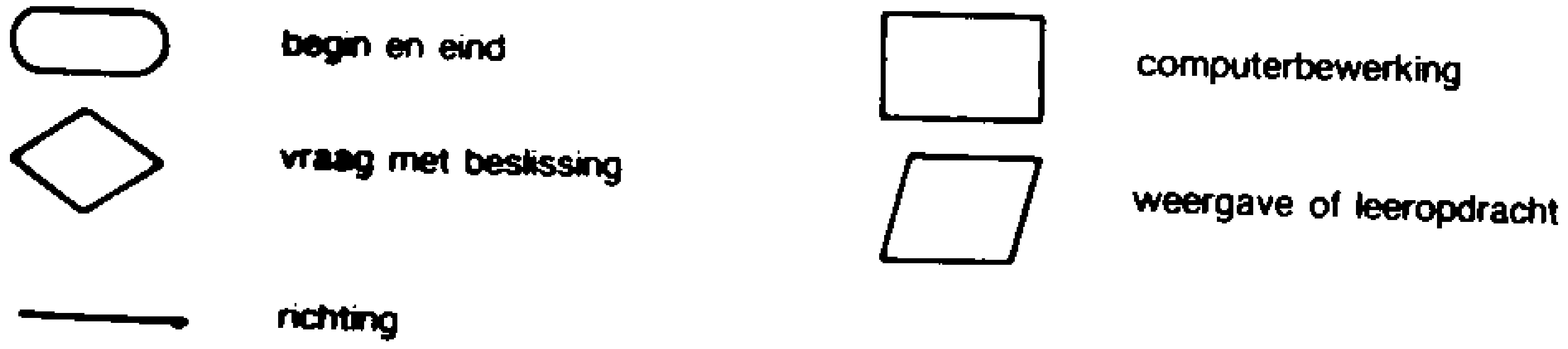
Na het indrukken van RETURN is de oude regel 20 vervangen door de nieuwe regel. Niets wijst daar echter op. Om de verandering te controleren kunt u LIST tikken.
Door het nummeren van de regels zijn deze aanwijsbaar en herkenbaar geworden. We kunnen nu telkens een regel veranderen. We kunnen ook verwijzen naar een bepaalde regel, maar ook binnen een programma kan worden verwezen. Neem het eerste programma van dit hoofdstuk, maar verander regel 20 net als hiervoor en voeg toe:

```
25 LET A=A-25
60 GOTO 25
```

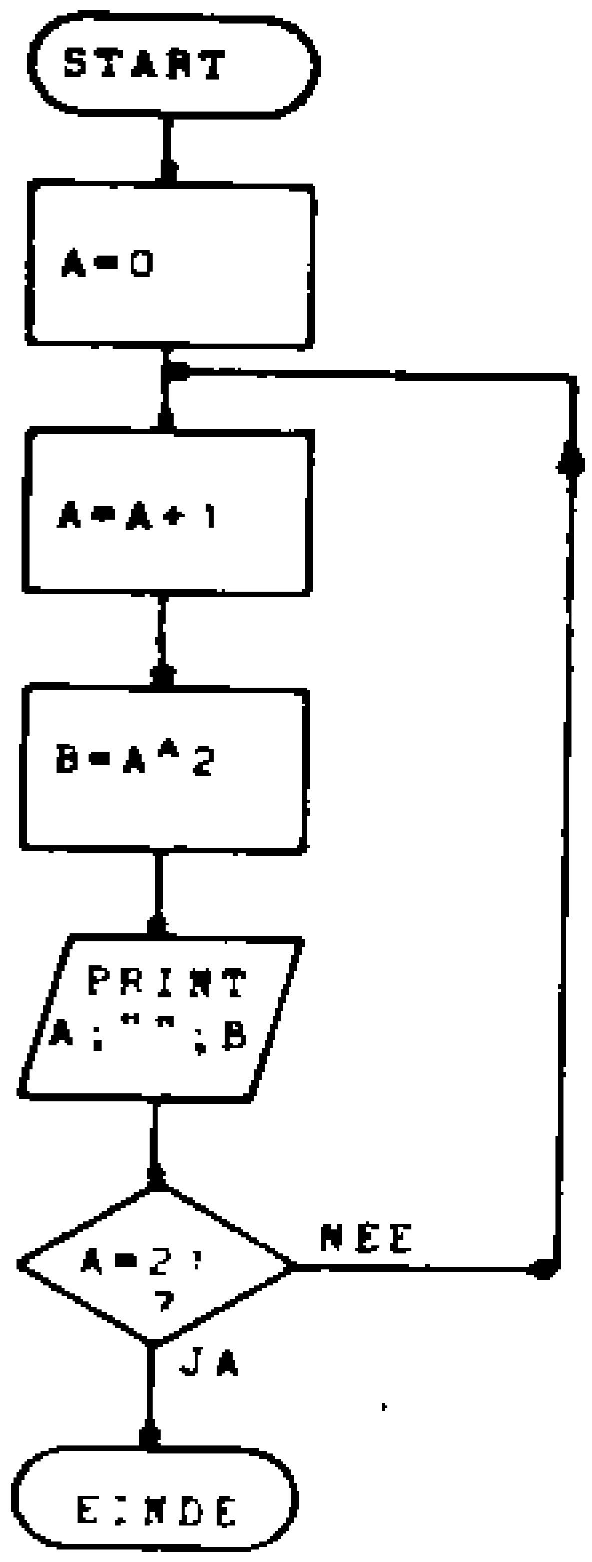
Laat het programma lopen (RUN, RETURN). De Atari geeft een lange lijst sommen met uitkomsten weer. U geeft in regel 60 immers de opdracht om weer terug te gaan naar regel 25 (GOTO = ga naar). Daar aangekomen vervolgt de Atari zijn weg door het programma van laag regelnummer naar hoog regelnummer, om uit te komen bij regel 60, die hem weer terugstuurt naar regel 25. Om het programma te stoppen gebruikt u RESET. Om het programma even stil te zetten gebruikt u net als bij de listing CONTROL en 1. In het Engels zijn 'go' en 'to' twee aparte woorden. Atari-BASIC maakt geen onderscheid tussen 'GOTO' en 'GO TO'.
Door de regelnummers zijn zogenaamde *'lussen'* mogelijk geworden. Een lus is een programma-onderdeel dat door de computer verschillende keren wordt uitgevoerd. U moet dit heel letterlijk opvatten. De computer loopt een bepaalde route door een programma. Bij een eenvoudig programma is die route een rechte weg. Zodra er echter lussen in het programma voorkomen, wordt de computer gedwongen een route te volgen die hem na verloop van tijd weer terugbrengt op een punt waar hij al eerder is geweest. De computer loopt in het programma een cirkel of lus. Als we bovenstaand programma netjes beschrijven in een stroomdiagram (een schema dat de loop of de stroom van een programma weergeeft) krijgen we:



Dit is een zeer eenvoudig stroomdiagram. De computer kan alsmaar recht uit lopen en slechts op een plaats wordt hij gedwongen om een boog te beschrijven zodat hij weer vlak bij zijn uitgangspunt uitkomt.
Het lijkt alsof het wat overdreven is om voor zo'n eenvoudig programma een hele route-kaart in de vorm van een stroomdiagram te maken. Zodra u echter iets ingewikkelder problemen aan wilt pakken, is het maken van een stroomdiagram erg handig. Alleen door een stroomdiagram raakt de programmeur de weg die de computer door het programma moet nemen niet kwijt.
De belangrijkste tekens die in een stroomdiagram worden gebruikt zijn:



Als voorbeeld het volgende probleem: Maak een lijst van de kwadraten van alle getallen van 1 tot 21. Een kwadraat van een getal is dat getal met zichzelf vermenigvuldigd. Het stroomdiagram kan er dan zo uitzien:



Als dit stroomdiagram wordt omgezet in een programma krijgt u onderstaand resultaat.
Het oude programma krijgt u uit het geheugen van de Atari door de direkte opdracht

NEW (=nieuw)

Door deze opdracht worden oude programma's en alle gegevens uit het geheugen van de Atari gehaald. U kunt nu met een schone lei beginnen. Test dit uit door na het intikken van NEW de opdracht LIST te geven. De computer geeft niets weer, maar geeft door middel van de melding READY weer dat de opdracht wel is vervuld. Er viel echter niets te listen, want het programma is door NEW verwijderd.
Een ruwere methode is het uit- en weer aanzetten van de computer. Dit zorgt echter voor onnodige slijtage.

```
10 REM kwadraten van 1 tot en met 20
20 LET A=0
30 LET A=A+1
40 LET B=A^2
50 PRINT A;" in het kwadraat is ";B
60 IF A<21 THEN GOTO 30
```

In regel 60 laten we de Atari testen of A nog steeds kleiner is dan 20 (A < 20). Zo nee, dan is A gelijk aan 20 en is het kwadraat daarvan net uitgerekend. Zo ja, dan moet het programma nog een keer worden doorlopen en wordt de machine terug gestuurd.

Het gebruik van stroomdiagrammen is geen overbodige luxe voor mensen die heel precies zijn. Zolang u korte eenvoudige programma's schrijft lijkt het stroomdiagram zo vanzelfsprekend, dat het opschrijven ervan overbodig is. Doet u het toch. De oefening die u opdoet met het maken van stroomdiagrammen voor eenvoudige programma's, heeft u nodig voor het maken van stroomdiagrammen van ingewikkelder programma's. Probeert u bij elk door u gebruikt programma in dit boek zelf een stroomdiagram te construeren.

Maakt u vooral veel gebruik van de opdracht **REM** (REMARK = opmerking). Hiermee kunt u in de listing van uw programma opmerkingen maken ter verduidelijking van het programma. Alle tekst na een REM-opdracht wordt door de computer genegeerd. Probeer dus niet na een REM-opdracht een andere opdracht in dezelfde programmaregel te zetten .

10 REM wis het scherm : GRAPHICS 0	Dit kan niet. De computer negeert de CLS-opdracht.
10 GRAPHICS 0 : REM wis het scherm	Dit kan wel. De computer wist het scherm.

Let op de methode waarop meer dan een opdracht in een programmaregel gezet wordt: **de opdrachten worden onderling gescheiden door een dubbele punt** (:).
U kunt het woord REM ook vervangen door een R met een punt erachter. De meeste Atari-BASIC opdrachten kennen een afgekorte vorm. Een lijst daarvan vindt u in **bijlage 3**. Voor de leesbaarheid is in de listings in dit boek telkens de volledige opdracht gegeven.
Merk op dat als u de opdracht in afgekorte vorm intikt, de listing toch het volledige woord weergeeft.

Hierboven zag u een klein programma met één enkele vertakking. Hieronder vindt u een programma met iets meer vertakkingen. Het programma zet graden Fahrenheit om in graden Celcius, of omgekeerd. De gebruikte formules zijn:

$$F = 9/5 * C + 32$$
$$C = 5/9 * (F - 32)$$

Het programma wordt:

```
10 REM graden omzetten
15 GRAPHICS 0
20 PRINT :PRINT "Dit programma zet graden F om in"
30 PRINT "graden C of andersom"
40 PRINT :PRINT
50 PRINT "Wilt u van F naar C (0)"
60 PRINT "of van C naar F      (1)"
70 PRINT :PRINT "Tik 0 of 1, RETURN"
80 REM keuze
90 INPUT X
100 IF X=0 THEN GOTO 180
110 REM als X=1 dan gaat prog. hier verder
120 PRINT "Hoeveel graden? ";:INPUT C
130 LET F=(9/5)*C+32
140 PRINT :PRINT C;" gr. Celc.= ";F;" gr. Fahr."
150 PRINT
160 REM de twee delen komen bij 230 bij elkaar
170 GOTO 230
180 REM als X=0 dan gaat prog. hier verder
190 PRINT "Hoeveel graden? ";:INPUT F
200 LET C=(5/9)*(F-32)
210 PRINT :PRINT F;" gr. Fahr.= ";C;" gr. Celc."
220 REM nu volgt automatisch 230
230 PRINT :PRINT "Wilt u nog een keer?"
240 PRINT "Tik 1 (ja) of 0 (nee), RETURN";:INPUT Y
250 REM na keuze 1 (ja) terug naar begin
260 IF Y=1 THEN PRINT "}":GOTO 40
270 REM bij keuze 0 (nee), einde
280 PRINT :PRINT
290 PRINT "Tot de volgende keer!"
```

Tik dit programma in en zet het daarna op een cassettebandje. Volg daarvoor de instrukties uit de handleiding. Tik:

CSAVE (bewaren)

In de programmarecorder *(de speciale Atari-cassetterecorder)* moet u een leeg stuk van een cassetteband klaar hebben staan. Nadat u de bovenstaande opdracht heeft ingevoerd door RETURN, piept de Atari tweemaal. Dat betekent dat u twee toetsen op de programmarecorder moet indrukken: **De PLAY- en de RECORD- knop moeten tegelijk worden ingedrukt.**
Druk nu op een willekeurige toets op de Atari, behalve de BREAK of een van de speciale funktietoetsen.
De programmarecorder gaat nu lopen en uw programma wordt net als een

muziekstuk op het bandje opgenomen. Als u het bandje ooit op een gewone cassetterecorder zou afdraaien zult u horen dat ook uw programma als een muziekstukje klinkt. Het is alleen niet zulke mooie muziek. Het computersignaal bestaat uit een hoge fluittoon met allerlei piepjes er doorheen. Zodra het gehele programma is opgenomen, stopt de motor van de programmarecorder en verschijnt er READY op het beeldscherm.
Uw programma staat nu veilig op cassette. Om dit te controleren tikt u eerst NEW. Uw programma bevindt zich niet meer in het geheugen. Overtuig u daarvan door LIST in te tikken.
Spoel de cassette nu terug naar de plaats waar het programma op de cassette werd opgenomen. Tik nu:

CLOAD (laden)

Nadat u op RETURN gedrukt heeft, geeft de computer één piepje. U moet nu slechts een knop van de programmarecorder indrukken, namelijk de PLAY-knop (van afspelen). Na het indrukken van de knop op de programmarecorder drukt u op een willekeurige toets (behalve BREAK of een funktietoets) van de computer. Het programma wordt nu in het geheugen van de computer geladen. Zodra de computer daarmee klaar is, verschijnt weer de READY-mededeling. Om te zien of het programma goed in het geheugen terecht is gekomen, kunt u LIST of RUN tikken.

Bij het opnemen en weer afspelen van programma's is het handig gebruik te maken van de bandteller van de programmarecorder. Het is erg vervelend als u programma's door elkaar heen gaat gooien of, erger, als u een nieuw programma opneemt over een ouder programma dat u nog wilt gebruiken. Om dit te voorkomen kunt u het beste elk programma op een 'ronde' tellerstand laten beginnen. Bijvoorbeeld: het eerste programma begint op tellerstand 000, het tweede programma begint op tellerstand 100, het derde programma begint op tellerstand 200, enzovoorts. Als u erg lange programma's heeft, zult u de tussenruimte zelfs groter moeten maken. Bewaar bij elke cassette een papiertje met daarop de programma's en de tellerstanden van de programma's op die cassette.
Een mooiere, maar duurdere oplossing, is om elk programma op een aparte (vaak korte) cassette te zetten. Nadeel daarvan is dat u al snel een enorme hoeveelheid cassettes heeft.

Er is nog een andere manier van het bewaren van een programma op een cassette. U gebruikt daarvoor de opdracht:

LIST "C:"

Deze opdracht zet het programma op een speciale manier op de cassette. Hierdoor kan dit programma later gemengd worden met een ander programma, dat zich reeds in het geheugen van de computer bevindt.
Het laden van een op zo'n wijze opgenomen programma moet ook op een speciale manier gebeuren:

ENTER "C:"

Deze opdracht leest een programma van de cassette en voegt dat bij het reeds in het geheugen aanwezige programma. Dit is gelijk aan de funktie MERGE bij sommige andere merken computers. Het programma in het geheugen van de computer wordt bij de CLOAD opdracht dus gewist, terwijl het programma bij een ENTER opdracht in het geheugen blijft. Let op! *Als beide programma's (in het geheugen en van cassette te laden programma) regelnummers hebben die hetzelfde zijn, worden van die programmaregels die gelijke nummers hebben, de programmaregels die al in het geheugen aanwezig waren, vervangen door de programmaregels met overeenkomende nummers van de cassette.* Het beste is dat u er zelf op let dat de twee samen te voegen programma's allebei verschillende regelnummers hebben. Bijvoorbeeld het eerste programma alleen regelnummers onder de 1000 en het andere programma alleen regelnummers boven de 1000.

Disk gebruikers

Voor gebruikers van een diskdrive geldt bijna hetzelfde verhaal. Alleen moet een disk gebruiker een naam aan elk programma geven. Het is niet mogelijk om programma's zonder naam op de diskette te zetten. *Gebruikt u een naam die al op de diskette voorkomt, dan wordt het programma met dezelfde naam op diskette gewist!*
U kunt geen gebruik maken van de CLOAD en CSAVE opdrachten. U kunt wel gebruiken:

LOAD "D:xxxxx"

en

SAVE "D:xxxxx"

of

LIST "D:xxxxx"

en

ENTER "D:xxxxx"

Op de plaats van de X-en komt de naam van het programma te staan. De naam kan maximaal 8 letters of cijfers lang zijn, moet met een letter beginnen en mag geen spaties bevatten.
Let u op het gebruik van de **D:**. Dit geeft aan dat er geen Cassetterecorder, maar één Diskdrive in gebruik is. Gebruikt u meer dan een diskdrive, dan dient u in de opdracht het nummer van de gewenste diskdrive op te geven.

LOAD "D1:xxxxx"

of

SAVE "D2:xxxxx"

Heel belangrijk voor diskgebruikers is dat zij elke keer VOORDAT ze met programmeren in BASIC beginnen **EERST de DOS** laden. Heeft u dat niet

gedaan, dan is het onmogelijk om een programma op een diskette te zetten! Vergeet u ook niet dat het onmogelijk is diskettes te gebruiken die niet geformateerd zijn. U kunt uw diskette formateren voor gebruik met een Atari door het op de DOS-diskette aanwezige formateringsprogramma. Zie ook **bijlage 6.**

Een lijst van alle programma's

Diskgebruikers kunnen een lijst opvragen met alle programma's die op een bepaalde diskette staan. Daarvoor zet u eerst het programma dat nog in het geheugen zit en bewaard moet blijven op de diskette met behulp van SAVE of ENTER.
De gebruikelijke methode is dat u nu de diskette uit de diskdrive haalt en de masterdiskette in de drive plaatst. Tik DOS en RETURN zodat het disk operating systeem wordt geladen. Het menu verschijnt en u kiest voor F (FILES INDEX) bij DOS 2.0 of DOS 3.0 of u kiest A (DISK DIRECTORY) bij DOS 2.5. Stop nu de diskette waarvan u een inhoudsopgave wilt hebben in de diskdrive. De computer vraagt om een FILESPECIFICATION (= aanduiding van de bestanden). Om alle programma's en bestanden op de schijf te zien tikt u nu:

```
*.*
```

Dit heet een *'wild-card'*. Normaal gesproken staat in de naamsaanduiding van een programma de naam voor de punt en achter de punt een soortaanduiding. Die soortaanduiding kunt u zelf vaststellen. Gebruikelijk is .BAS voor BASIC programma's en .TXT voor tekstbestanden van uw tekstverwerker. Overal waar de computer in de filespecificatie een asterisk tegenkomt, beschouwt hij deze als een soort 'joker'. Op die plaats mag alles ingevuld worden. Als u *.* invult mag de computer alle programma's en bestanden lezen om in de inhoudsopgave te zetten. De naam is vrij (*) en het soort bestand/programma is ook vrij (.*).
Om een inhoudsopgave van alleen de BASIC-programma's te krijgen, tikt u:

```
*.BAS
```

Dit gaat natuurlijk alleen als u al uw BASIC programma's altijd een naam heeft gegeven die eindigt met .BAS.

Het nadeel van bovengenoemde methode is dat u telkens van schijf moet wisselen en DOS opnieuw moet laden. Er bestaat een kleine truuk om dit te omzeilen. Met behulp van een één-regelig BASIC programma is het mogelijk de inhoudsopgave van een diskette te lezen. Stop de gewenste diskette in de drive en tik:

```
OPEN#1,6,0,"D:*.*":FOR X=1 TO 1E9:GET#1,A:PRINT
CHR$(a);:NEXT X
```

Hoe dit programma werkt is nu niet zo belangrijk. Wel belangrijk is dat dit programma een keurige lijst van alle op de diskette aanwezige programma's geeft, zonder dat DOS opnieuw geladen hoeft te worden. Na de inhoudsopgave krijgt u foutmelding 136 te zien. Het is wel zo netjes om deze weg te werken door een END opdracht (=einde opdracht). Hiermee worden alle kanalen naar de verschillende randapparaten gesloten, zodat de computer zich daar niet meer mee kan vergissen.

Heeft u bepaalde programma's die u niet meer op de schijf wilt hebben dan kunt u deze verwijderen door middel van de ERASE (wis uit) of DELETE (verwijder) funktie van DOS. Laad DOS en kies een van voorgenoemde mogelijkheden uit het menu. De computer vraagt weer om een filespec. Denk eraan dat als u nu een wildcard gebruikt, er verschillende programma's gewist zullen worden. **Tik dus de volledige naam in van het programma dat u wenst te wissen.**
De computer zoekt het programma op, plaatst de naam voor u op het scherm en vraagt of u het zeker weet. U moet een Y (yes) tikken als u dat programma wilt verwijderen. Heeft u het programma eenmaal verwijderd, dan is er geen weg meer terug!

Het gradenprogramma

In het gradenprogramma zitten enkele punten die het waard zijn om nog even op terug te komen.
De INPUT en IF..THEN.. opdrachten zijn nog niet eerder in dit boek aan de orde gekomen. De funktie van INPUT is de gebruiker zelf een getal, getallen, letter of letters in te laten tikken.
Door middel van IF..THEN.. springt de computer afhankelijk van een bepaalde voorwaarde naar een ander deel van het programma. Op deze manier kunnen we de twee keuzemogelijkheden van het programma (F en C en 'nog een keer') opgeven aan de computer. Omdat het hier echter alleen om het vertakken (met daarbij een leuk programma) ging, verwijzen we u voor de uitleg van deze instrukties naar de komende hoofdstukken.
Nog een enkele tip: Door een enkele PRINT-opdracht (in uw eigen listings af te korten tot een vraagteken) krijgt u een beter leesbaar scherm. Daarbij zorgt een PRINT opdracht voor extra ruimte in de programmalisting, waardoor deze beter leesbaar wordt.

5. VARIABELEN

Dit hoofdstuk is vooral bedoeld voor diegenen die weinig of geen wiskundige kennis bezitten. Voor hen geeft dit hoofdstuk een summiere uitleg van het gebruik van variabelen, onmisbaar bij het programmeren. Weet u hier voldoende van af, leest u dit hoofdstuk dan vluchtig door, er staan enkele wetenswaardigheden in met betrekking tot het verschil in het gebruik van variabelen in de wiskunde en bij het programmeren.

Over rekenen heeft u al eerder in dit boek kunnen lezen. Enkele voorbeelden:

2 + 2 = 4
6 - 3 = 3
2 x 2 = 4
6 : 2 = 3

Met deze manier van rekenen valt, ook door middel van de computer, veel uit te rekenen. Maar het wordt wat lastig als bepaalde getallen in een programma telkens veranderen. Elke keer zou daarvoor het programma geLIST en de getallen veranderd moeten worden.

De som 5 X 4 = 20 is goed uitgerekend, maar het is niet interessant om dit door de computer uit te laten rekenen. De som wordt al een stuk interessanter als u weet dat de getallen eigenlijk staan voor:

Lengte X Breedte = Oppervlakte

Met de hier gegeven 'formule' is het mogelijk veel meer getallenvoorbeelden te gebruiken. De eerste som zegt alleen maar dat 5 maal 4 gelijk is aan 20. De tweede som vertelt veel meer. Zo geldt bijvoorbeeld dat van elke rechthoek de lengte maal de breedte gelijk is aan de oppervlakte. Een rechthoek met een lengte van 5 en een breedte van 4 heeft een oppervlakte van 20. Wat hebben we hier gedaan? De woorden Lengte, Breedte en Oppervlakte zijn tijdelijk vervangen door 4, 5 en 20. Er kunnen voor dezelfde woorden ook andere getallen ingevuld worden:

6 (Lengte) X 2 (Breedte) = 12 (Oppervlakte)

Als we nu de som in de vorm Lengte X Breedte = Oppervlakte aan de computer opgeven en vervolgens telkens aan de computer vertellen hoe groot de Lengte en hoe groot de Breedte is, kan de computer telkens opnieuw de Oppervlakte van een rechthoek uitrekenen. Op deze wijze zijn de vaste getallen vervangen door 'getallen' die kunnen varieren, veranderen (zogenaamde **VARIABELEN**). De formule kan nu algemeen gebruikt worden. Er hoeft nu immers maar één keer aan de computer opgegeven te worden wat er met de variabelen moet gebeuren, waarna er telkens alleen de nieuwe waarden voor de variabelen opgegeven hoeven te worden.

Lengte X Breedte = Oppervlakte

De computer krijgt opgegeven dat Lengte=5 en Breedte=4. De computer vult de getallen in. In zijn geheugen staat nu:

5 X 4 = Oppervlakte

Als de computer om het resultaat van de vergelijking gevraagd wordt, zal hij 20 (Oppervlakte = 20) antwoorden. Het grote voordeel van het rekenen met variabelen in plaats van vaste getallen is dat variabelen op een eenvoudige wijze ook andere waarden kunnen krijgen. Zo kan de computer ook worden opgegeven dat Lengte=7 en Breedte=5. De computer vult de vergelijking weer in en in het geheugen staat dan:

7 X 5 = Oppervlakte

Net zoals u kunt rekenen met getallen kunt u ook rekenen met variabelen. Dit heet dan algebra.

Nog een voorbeeld van het gebruik van variabelen. Laten we aannemen dat u elke week een kiosk binnen stapt om daar een aantal kranten en tijdschriften te kopen. Elke krant kost 2 gulden en elk tijdschrift kost 5 gulden. De prijs die u aan de kassa moet betalen is natuurlijk afhankelijk van het aantal kranten en het aantal tijdschriften dat u koopt. Voor 3 kranten en 4 tijdschriften betaalt u:

3 X 2 gulden + 4 X 5 gulden = 26 gulden

Voor 4 kranten en 7 tijdschriften betaalt u:

4 X 2 gulden + 7 X 5 gulden = 43 gulden

Als u met de computer wilt uitrekenen hoeveel u moet betalen hoeft u niet elke keer de hele rekensom in te tikken. U vertelt de computer met welke formule hij die prijs moet uitrekenen en verder tikt u alleen iedere week in, welk aantal kranten en welk aantal tijdschriften u wilt kopen. De computer rekent dan zelf de prijs uit. U geeft de computer de volgende formule op:

prijs = aantal kranten X 2 + aantal tijdschriften X 5

of wat korter:

PR = K X 2 + T X 5

Als u vervolgens de prijs wilt weten van 3 kranten en 4 tijdschriften, geeft u aan de computer op: K=3 en T=4. De computer vult vervolgens zelf de waarden in: PR = 3 X 2 + 4 X 5 en geeft als antwoord: PR = 26. K en T zijn variabelen: we kunnen hiervoor steeds een ander getal invullen.

Let bij deze voorbeelden op het vermenigvuldigingsteken. Dit wordt bij computers aangegeven met een **asterisk (*)**.

Enkele punten vragen bij het gebruik van variabelen wel speciale aandacht Bij het programmeren mag niet zo maar elke naam aan een variabele gegeven worden. Er mogen geen spaties of leestekens in de naam voorkomen. De naam kan net zoveel karakters bevatten als het geheugen van uw Atari lang is (bij de 130 XE kunt u niet zo maar gebruik maken van de 4 x 16 K extra). Het is echter goede gewoonte de naam niet langer te maken dan **130 tekens**. Dit is de maximale lengte van de invoerbuffer. Dat is de ruimte die de Atari gebruikt om binnenkomende gegevens tijdelijk in op te slaan. De letters moeten allemaal **HOOFDLETTERS** zijn. De tweede letter van een variabelenaam mag elk letterteken zijn, maar de eerste letter moet een letter uit het alphabet zijn. Binnen een variabelenaam mag geen **BASIC-sleutelwoord** voorkomen. BASIC-sleutelwoorden zijn alle woorden die in BASIC een funktie vervullen. Zo is bijvoorbeeld de variabelenaam *TOETER* niet toegestaan omdat deze begint met het BASIC-woord *TO*.
In de algebra mag u het vermenigvuldigingsteken bij het gebruik van variabelen weglaten. Bij het programmeren mag dit niet. De computer beschouwt 'AB' als de variabele met de naam 'AB', en 'A*B' als het produkt van de variabele 'A' met de variabele 'B'.
In Atari-BASIC kunt u in een programma maximaal **128** verschillende variabelen gebruiken.
U kent een waarde aan een variabele toe met behulp van de BASIC-opdracht

LET

Bijvoorbeeld LET B = 5. *Hierdoor wordt aan de variabele 'B' de waarde '5' toegekend.*
Daarnaast kan aan een variabele ook een waarde toegekend worden tijdens het RUNnen van het programma door verschillende, later in dit boek te behandelen opdrachten, zoals:

INPUT

Bijvoorbeeld INPUT B. Hierdoor wordt aan de variabele 'B' de waarde toegekend, die de gebruiker na het verschijnen van een vraagteken op het beeldscherm intikt en invoert, door middel van RETURN.
Een variabele kan ook een variabele waarde toegekend krijgen door middel van LET. Bijvoorbeeld LET A = B + 3. Hierdoor krijgt de variabele 'A' de waarde 'B + 3'. Let op! Dit is alleen mogelijk als de computer al weet wat de waarde van 'B' is. Als u de computer niet van te voren een waarde van B opgeeft gaat de computer ervan uit dat u de waarde 0 (nul) bedoelt. Tikt u maar:

```
LET A=Q+10
```

Als u de variabele Q niet eerder tijdens het aanstaan van de computer, een waarde gegeven heeft, zal het antwoord van de opdracht:

PRINT A

gelijk zijn aan 10. Dat is gelijk aan 0 (voor Q) + 10. Het is beter om er voor te zorgen dat een variabele altijd al een waarde toegekend heeft gekregen, voordat u hem gebruikt. Het leidt anders gemakkelijk tot fouten. U kunt dit doen ofwel door middel van een direkte opdracht, ofwel doordat de waardetoekenning aan 'Q' plaatsvindt in het computerprogramma voor de waardetoekenning aan 'A'. De meeste andere microcomputers staan u helemaal niet toe een variabele te gebruiken voordat er een waarde aan is toegekend. Zou u het bovenstaande voorbeeld op bijvoorbeeld een BBC computer doen dan zou u een foutmelding krijgen die u zegt dat de variabele niet bestaat. Aan Q is nog geen waarde toegekend. Zolang er aan een variabele geen waarde is toegekend, bestaat de variabele voor andere dan Atari-computers niet. Omdat de Atari u wel toestaat om een variabele te gebruiken die nog geen waarde heeft, en deze variabele zelf de waarde 0 geeft, zult u zelf extra goed moeten opletten.

Bij het toekennen van waarden aan variabelen moet het gelijkteken (=) niet beschouwd worden als deel uitmakende van een bewering:

LET A = A + 1

Dit dient gelezen te worden als: de waarde van de variabele 'A' is vanaf nu gelijk aan de oude waarde van 'A' plus 1. Of anders gezegd: tel bij de waarde van de variabele 'A' een op en laat het resultaat de nieuwe waarde van 'A' zijn. Het moet niet gelezen worden als 'A' is gelijk aan 'A+1'.
Dit is immers een onware bewering.
Het woordje LET is in Atari-BASIC niet verplicht om de waarde van een variabele op te geven. U kunt dus ook tikken:

A = 10

De waarden van variabelen kunnen zowel getallen (numerieke variabelen) als reeksen van letters (stringvariabelen) zijn. Over de stringvariabelen kunt u meer lezen in hoofdstuk 9 over strings.

Sommige computers kennen drie soorten numerieke variabelen: variabelen waarin alleen gehele getallen opgeslagen kunnen worden (integers), en variabelen waarin reeele getallen kunnen worden opgeslagen, onderverdeeld in opslag met enkele of dubbele precisie.
In een integer variabele kan alleen een geheel getal opgeslagen worden. De minimale waarde die opgeslagen kan worden is -32768. De maximale waarde is 32767. Een integervariabele wordt meestal onderscheiden van andere numerieke variabelen door het procentteken (%) achter de naam van de variabele.
In een numerieke variabele met enkele precisie kan meestal een getal met maximaal 6 cijfers (voor of achter de komma) worden opgeslagen. Worden de getallen te groot, dan wordt het teveel aan cijfers afgerond en noteren de niet-Atari computers het getal in de vorm van een exponent met grondtal 10.

Bijvoorbeeld: A! = 9.69974E+17
Enkele precisie variabelen worden aangegeven met een uitroepteken (!) achter de naam van de variabele.
Numerieke variabelen met dubbele precisie worden aangegeven met een Amerikaans aantalsymbool of matje (#) achter de naam van de variabele. Deze variabelen kunnen getallen bevatten met maximaal 14 cijfers. Net als bij getallen met enkele precisie wordt het teveel aan cijfers afgerond, en het getal genoteerd met een E: A# = 9.6997445234213E+17

De Atari kent al deze onderscheidingen niet. Elk getal wordt in de Atari opgeslagen als een reeel getal. In een variabele kunt u dus zowel een geheel getal als een getal met verschillende cijfers achter de komma opslaan. U hoeft zich dus nooit het hoofd te breken over de vraag welke soort variabele u bij een bepaalde toepassing nodig heeft. Schrijft u echter programma's die ook voor andere merken computer gebruikt gaan worden dan zult u het bovenstaande wel in gedachten moeten houden. De notatieverschillen tussen de verschillende soorten variabelen kunt u gebruiken als u programma's die voor andere computers zijn geschreven (en waar de %, !, en # in voorkomen) wilt aanpassen voor gebruik op de Atari.

Het volgende voorbeeldprogramma geeft goed aan hoe u elk woord kunt gebruiken om een variabele te maken. De woorden zijn zo gekozen dat de werking van het programma duidelijker wordt. In dit programma worden lijnen van links naar rechts en van boven naar onder getrokken. Dat gebeurt een aantal keren. De gebruikte variabelenamen zijn dan ook LINKSRECHTS, BOVENONDER en TELLER.

```
10 TELLER=1
15 BOVENONDER=0
20 GRAPHICS 9
30 FOR LINKSRECHTS=1 TO 79
40 TELLER=TELLER+1:IF TELLER>9 THEN TELLER=1
50 COLOR TELLER
60 PLOT LINKSRECHTS,BOVENONDER:DRAWTO 79-LINKSRECHTS,BOVENONDER
70 PLOT LINKSRECHTS,BOVENONDER:DRAWTO 79-LINKSRECHTS,190-BOVENONDER
80 PLOT LINKSRECHTS,190-BOVENONDER:DRAWTO 79-LINKSRECHTS,190-BOVENONDER
90 PLOT LINKSRECHTS,190-BOVENONDER:DRAWTO 79-LINKSRECHTS,BOVENONDER
100 BOVENONDER=BOVENONDER+6:IF BOVENONDER>190 THEN BOVENONDER=0
110 NEXT LINKSRECHTS:TELLER=TELLER+1:GOTO 20
```

Tik het programma in, druk na elke regel op RETURN en tik aan het einde RUN en RETURN. De Atari laat nu op een fraaie en snelle manier zijn grafische mogelijkheden zien.
Het programma wordt wel duidelijker door het gebruik van lange variabelenamen (als u het programma over tien jaar nog een keer leest is het duidelijk waar het over gaat) maar het is lastig in te tikken. U kunt natuurlijk ook wel iets kortere aanduidingen gebruiken. Vooral bij grafische toepassingen is het gebrui-

kelijk om de richting links-rechts de naam X te geven. De richting bovenonder wordt dan Y. Voor een teller wordt in computerprogramma's heel vaak een I gebruikt. Mooier is het om een T te gebruiken. Een I lijkt immers op een 1.
Verander bovenstaand programma en kijk of het gemakkelijk in te tikken is.

```
10 T=1
15 Y=0
20 GRAPHICS 11
30 FOR X=1 TO 79
40 T=T+1
50 COLOR T
60 PLOT X,Y:DRAWTO 79-X,190-Y
70 PLOT X,Y:DRAWTO 79-X,Y
80 PLOT 79-X,Y:DRAWTO X,190-Y
90 PLOT X,190-Y:DRAWTO 79-X,Y
100 Y=Y+2:IF Y>190 THEN Y=0
110 NEXT X:GOTO 20
```

Dit programma is een kleine variant op het vorige. Probeer ook in regel 20, GRAPHICS 11 te veranderen in GRAPHICS 9.
In het eerste programma is gekozen voor BOVENONDER als variabelenaam. Merk op dat ONDERBOVEN als variabelenaam niet toegestaan is. Deze naam begint met ON. Dit is een onderdeel van een BASIC opdracht.

6. INVOER EN BEELDWEERGAVE

Input

Om te kunnen werken met de computer moeten we iets kunnen invoeren *(INPUT)*. Om te zien wat de computer doet of wat hij gedaan heeft, is het nodig dat de computer mededelingen kwijt kan *(OUTPUT)*. De Atari-computer doet dit meestal via een televisie of een monitor.
Een deel van de mogelijkheden van het invoeren van gegevens heeft u reeds kunnen lezen in de hoofdstukken 1 en 4. Wat echter nog niet aan de orde is geweest is het invoeren van gegevens terwijl het programma aan het lopen is (zoals dat heet: terwijl het programma runt - van het engelse werkwoord run = lopen). De mogelijkheid om gegevens in te voeren tijdens de run van een programma zou verschillende voordelen hebben. Het is niet nodig dat bij elke gegevensverandering het programma wordt stilgezet om het gegeven in te voeren, waarna het programma opnieuw gestart moet worden. Bij het spelen van spelletjes is het zelfs onontbeerlijk, want er moet een wisselwerking tussen u en de computer zijn. U moet tijdens de programma-run kunnen reageren op opdrachten of signalen van de computer door bepaalde gegevens op te geven. Atari-BASIC kent hiervoor de opdracht INPUT (=invoer). Tik:

```
NEW
10 INPUT A
20 PRINT A;A*A
```

RUN dit programma (tik RUN gevolgd door RETURN). Er verschijnt een vraagteken links in beeld met daarachter de cursor. De Atari vraagt u een waarde in te tikken. Tik 12 (uiteraard gevolgd door RETURN).
Op het beeldscherm verschijnt het cryptische getal 12144. U had natuurlijk al begrepen dat dit twee getallen zijn: de waarde van A, waarvoor u zelf 12 ingetikt hebt, en de waarde van A*A=144. Doordat in regel 20 tussen A en A*A een punt-komma (;) is geplaatst, zet de Atari de getallen direkt achter elkaar zonder tussenruimte. In dit geval niet zo fraai, maar een faciliteit waarvan u nog vaak gebruik zult maken.

De INPUT-opdracht werkt vergelijkbaar met de LET-opdracht. Door de INPUT-opdracht wacht de computer totdat u een waarde intikt. Deze waarde wordt toegekend aan de variabele genoemd achter de INPUT-opdracht. Het is ook mogelijk verschillende variabelen een waarde toe te kennen door middel van INPUT. Tik:

```
10 INPUT A,B,C
20 PRINT A;"*";B;"*";C;"=";A*B*C
```

De asterisken tussen de aanhalingstekens worden niet 'uitgerekend' door de computer, maar dienen om netjes tussen de getallen op het scherm te worden

gezet. Ze staan tussen aanhalingstekens zodat de computer er niet mee gaat rekenen.
Als u dit programma runt verschijnt er weer een vraagteken met daarachter de cursor. U kunt nu drie getallen invullen. De getallen die u opgeeft kunt u op twee manieren scheiden: door komma's of door RETURN's. In beide gevallen moet de laatste opdracht een RETURN zijn. U kunt dus tikken:

```
    1,2,3   (RETURN)
of
    1       (RETURN)
    2       (RETURN)
    3       (RETURN)
of
    1,2     (RETURN)
    3       (RETURN)
of
    1       (RETURN)
    2,3     (RETURN)
```

U ziet dat het er niet veel toe doet, op welke manier u de verschillende getallen opgeeft. De enige eis die wordt gesteld, is dat de getallen gescheiden zijn, door een komma of door een RETURN.
U moet altijd het juiste aantal gegevens invoeren. Dat wil zeggen net zoveel gegevens als de computer voor het programma nodig heeft. Als u te weinig gegevens invult, zal de computer alleen een vraagteken laten zien en wachten totdat u meer gegevens invult. Vult u teveel gegevens in, dan worden de extra gegevens verwaarloosd. Vul in bovenstaand programma eens 1,2,3,4 in. De computer meldt niet eens dat hij gegevens weglaat.
Spaties die u (per ongeluk) voor of achter het getal invoert, worden door de computer genegeerd. Probeert u als voorbeeld de cijfers 2, 20,30 . Spaties binnen een getal leiden tot een foutmelding: Probeer 2, 20, 3 10.

Berichten

Er is een nadeel verbonden aan het op bovenstaande manier gebruiken van de INPUT-opdracht. Bij een klein programma als het bovenstaande is nog wel duidelijk wat er ingevuld moet worden door de gebruiker als de computer een vraagteken op het scherm weergeeft. Maar bij een langer programma is het bepaald niet denkbeeldig dat de gebruiker vergeet wat de computer precies voor een getal als invoer wil. En wat te denken van de niets vermoedende kennis, die u een zelf gemaakt programma voorschotelt. Die komt er helemaal niet meer uit.
Het is daarom nodig dat u in het programma voorzieningen treft die ervoor zorgen, dat de gebruiker weet wat voor een soort getal (of misschien wel een stel lettertekens) hij of zij moet invullen. Verschillende merken computers staan met hun BASIC toe dat u een 'berichtje' bij de INPUT opdracht plaatst. U zult dit vaak tegenkomen bij listings voor die computers in de vorm: INPUT

"hier berichtje" ; A. In dit voorbeeld is A de variabele die een waarde krijgt door het invullen van een waarde na het INPUT vraagteken. Voor het vraagteken wordt het opgegeven bericht weergegeven.
Deze faciliteit heeft Atari-BASIC niet. Maar geen nood, met weinig werk is precies hetzelfde effect te bereiken. U dient echter ook de andere methode te kennen. U heeft het al vaker gehoord: bij het schrijven voor of het omzetten van programma's van andere computers dient u ook iets te weten van de BASIC zoals die bij andere computers werkt.
Een bericht bij een INPUT opdracht geeft u door een PRINT opdracht onmiddellijk voor de INPUT opdracht.

```
10 PRINT "Voer een getal in";
20 INPUT A
30 PRINT A;"*";A;"=";A*A
```

Op deze wijze wordt de gebruiker gevraagd om een getal in te voeren. Let op de **punt-komma achter** de PRINT opdracht. Zoals eerder vermeld, zorgt zo'n punt-komma ervoor dat de volgende tekst die het programma op het scherm afdrukt, direkt achter de vorige tekst wordt geplaatst. Dat wil zeggen dat het vraagteken van de INPUT opdracht keurig achter de afgedrukte tekst van de INPUT opdracht komt te staan.
Het volgende programma laat zien hoe u berichten bij INPUT kunt gebruiken. Alleen de eerste 70 regels zijn voor dit hoofdstuk van belang. Door het hele programma in te tikken en te laten lopen, ziet u echter tevens de snelheid waarmee de Atari kan tekenen, zowel in kleur als zwart/wit.

```
10 GRAPHICS 0
20 PRINT "HOEVEEL LIJNEN";
30 INPUT LN
40 PRINT :PRINT "ZWART/WIT (0) OF KLEUR (1)";
50 INPUT KL
60 PRINT :PRINT "LIJN (0) OF RECHTHOEK (1)";
70 INPUT V
80 IF KL=0 THEN GRAPHICS 9
90 IF KL=1 THEN GRAPHICS 11
95 IF V=1 THEN GOTO 200
100 FOR T=1 TO LN
110 COLOR T
120 X=INT(RND(0)*79)
130 Y=INT(RND(0)*191)
140 Q=INT(RND(0)*79)
150 R=INT(RND(0)*191)
160 PLOT X,Y:DRAWTO Q,R
170 NEXT T
180 END
200 FOR T=1 TO LN
210 COLOR T
```

```
220 X=INT(RND(0)*79)
230 Y=INT(RND(0)*191)
240 Q=INT(RND(0)*79)
250 R=INT(RND(0)*191)
260 PLOT X,Y:DRAWTO X,R
270 DRAWTO Q,R:DRAWTO Q,Y
280 DRAWTO X,Y
290 NEXT T
300 END
```

De opdrachten in regel 20 en 30 zorgen ervoor dat de gebruiker kan kiezen voor een bepaald aantal lijnen.
De regels 40 en 50 laten de gebruiker kiezen tussen kleur en zwart/wit. De regels 60 en 70 vragen de gebruiker of hij alleen lijnen wil zien, of rechthoeken. Nadat alle vragen door de gebruiker zijn beantwoord, gaat de computer aan de hand van de verkregen informatie verder met het programma. Afhankelijk van de keuzes van de gebruiker komt er een bepaald aantal lijnen of rechthoeken in kleur of zwart/wit op willekeurige plaatsen op het scherm. De computer kan ze bijzonder snel achter elkaar tekenen. Vergeet u niet dit programma na het intikken op een cassette of diskette op te slaan. Het programma is bij uitstek geschikt om kennissen te overtuigen van de snelheid en kleur van uw Atari.
De opdrachten die verder in het programma worden gebruikt zullen later in dit boek allemaal aan de orde komen.

Weergave op het beeldscherm

De keerzijde van het invoeren van gegevens, is de weergave door de computer van gegevens op het beeldscherm. Het weergeven van gegevens en berichten door de computer op het beeldscherm gebeurt meestal met de PRINT (= afdruk) opdracht. Het is u natuurlijk al opgevallen dat teksten die met behulp van deze opdracht worden afgedrukt, niet helemaal links in beeld beginnen, er blijft een kleine kantlijn bestaan. De plaats van de af te drukken tekst kan op verschillende manieren worden bepaald. Allereerst kunnen verschillende leestekens tussen de verschillende te printen gegevens worden geplaatst. Probeert u het volgende programma (tik eerst NEW om eventuele oude programma's te verwijderen uit het geheugen):

```
10 A=4
20 B=A^2
30 PRINT A;B;B^2
40 PRINT "ATARI";"MICRO";"COMPUTER"
```

Als u dit programma runt ziet u dat de teksten tegen elkaar aan worden gezet en de getallen samen één lang getal vormen.

RUN
416256
ATARIMICROCOMPUTER

Een heel ander effect krijgt u als u in plaats van de punt-komma een komma gebruikt om de verschillende af te drukken teksten en variabelen te scheiden:

```
10 A=4
20 B=A^2
30 PRINT A,B,B^2
40 PRINT "ATARI","MICRO";"COMPUTER"
```

Als u dit programma runt ziet u dat de verschillende items van elkaar gescheiden zijn.

```
RUN
4         16            256
ATARI     MICROCOMPUTER
```

Het beeldscherm is verdeeld in kolommen. Door de komma te gebruiken laat u de PRINT positie naar een volgende kolom verspringen. Dit is vergelijkbaar met het gebruik van de tabulatietoets van schrijfmachines.

Tabulatie

Zolang u gebruik maakt van de normale schermmodus, die aanstaat als u de computer aanzet, of die u aanroept met GRAPHICS 0, bestaat het scherm uit **24** regels van elk **40** lettertekens breed.
De posities van het scherm worden genummerd van 0 tot en met 39. Als u echter uw computer net heeft aangezet en u probeert om 40 lettertekens op een regel te zetten, zult u merken dat u niet verder komt dan 38 lettertekens. Probeert u op een willekeurige nieuwe regel maar, hoe vaak u de A in kunt drukken voordat de regel vol is.
Dit effect heeft te maken met de al eerder genoemde kantlijn van 2 lettertekens aan de linkerkant. Deze kantlijn zorgt ook bij het verspringen naar de volgende tabulatiestop voor vreemde resultaten.
Kijk wat er gebeurt als u verschillende malen achter elkaar de TAB toets indrukt. De cursor springt steeds een eindje verder, en wel steeds naar dezelfde posities op het scherm. De eerste positie is 5 plaatsen naar rechts, de andere posities zijn telkens 8 plaatsen naar rechts. U kunt deze plaatsen veranderen door de TAB toets te gebruiken *samen* met de CONTROL of de SHIFT toets. Plaats de cursor op de eerste TAB positie (dat wil zeggen vijf plaatsen vanaf de kantlijn naar rechts). Druk nu CONTROL en TAB tegelijk in en druk op RETURN. Als u nu weer op TAB drukt zult u zien dat de cursor vanaf de kantlijn niet vijf maar dertien plaatsen naar rechts verspringt. De eerste TAB positie is verwijderd. Voor het maken van een nieuwe TAB positie op een willekeurige plaats, zet u de cursor op de door u gewenste positie en drukt u de

TAB en de SHIFT toetsen tegelijk in, uiteraard gevolgd door RETURN. Bij het drukken op de TAB toets zal de computer de hierdoor aangegeven positie als TAB **stop** beschouwen. Het aan en uitschakelen van de computer zorgt ervoor dat de oorspronkelijke TAB posities weer worden gebruikt.
De komma als scheidingsteken tussen de verschillende PRINT items zorgt voor een TAB sprong. Deze TAB sprong is echter afwijkend van de sprong bij de TAB toets. Probeer het volgende programma uit.

```
10 A=4
20 B=A^2
30 PRINT A,A,A,A,A,A,A,A,A,A
40 PRINT "ATARI","MICRO";"COMPUTER"
```

Het resultaat:

```
RUN
4         4         4         4
 4         4         4         4
  4         4
ATARI     MICROCOMPUTER
```

De komma in een programma om te tabuleren zorgt niet voor het verder springen naar een vaste positie, maar voor het verder springen naar een positie een bepaald aantal plaatsen verderop. Als u sinds het aanzetten van de computer dit aantal nog niet heeft beinvloed, zal de cursor altijd 10 lettertekenposities verder springen door een komma tussen twee PRINT items.
Opvallend is hierbij dat hoewel het beeldscherm 40 lettertekenposities breed is, en er dus precies een veelvoud van 10 lettertekenposities op passen, de in het voorbeeld afgedrukt 4-en toch verspringen. Dit komt door de kantlijn van 2 die de Atari altijd aanhoudt. Deze kantlijn zorgt ervoor dat ook bij minder goed afgestelde televisies, waarbij het beeld een beetje naar links verschoven is, de tekst toch nog goed te lezen is. De programmeur dient er rekening mee te houden dat, zolang hij of zij geen maatregelen treft, het beeldscherm in de praktijk slechts 38 lettertekenposities breed is.
De afstand die de cursor verspringt door een komma in een PRINT opdracht, is te beinvloeden. U gebruikt daarvoor een POKE opdracht. Een POKE opdracht is een opdracht die een bepaalde waarde direkt in een opgegeven plaats in het geheugen zet. Alle plaatsen in het geheugen zijn genummerd, zodat ze aanwijsbaar zijn. Een bepaalde plaats in het geheugen wordt dan ook wel geheugenlokatie of geheugenadres genoemd.
De afstand die de cursor aflegt bij een komma in een PRINT opdracht onthoudt de computer als getal in geheugenadres 201. Door het getal dat zich in die geheugenlokatie bevindt te veranderen, kunnen we de computer iets anders laten 'denken'. We kunnen met een *POKE* opdracht dus direkt in het geheugen van de computer ingrijpen.

POKE 201,5

zorgt ervoor dat de cursor telkens slechts 5 posities verspringt. Tik deze direkte opdracht in en run het vorige programma opnieuw (door het veranderen van de waarde in een enkel geheugenadres wordt het programma -elders in het geheugen- niet beinvloed).
Ziet u dat de 4-en nu veel dichter bij elkaar komen te staan?
Ziet u dat de afstand tussen ATARI en MICROCOMPUTER niet is veranderd? En toch moet de cursor tussen het begin van ATARI en van MICROCOMPUTER slechts 5 posities verspringen. Telt u dat na, dan ziet u dat het woord MICROCOMPUTER direkt achter ATARI zou moeten beginnen. Dat is immers 5 lettertekenposities verder. Maar de computer is zo slim geprogrammeerd dat hij begrijpt dat u met een komma in een PRINT opdracht altijd een zekere scheiding tussen de items wenst. Komt het zo uit dat door het verspringen van de cursor de verschillende items toch direkt achter elkaar dreigen te komen, dan laat de computer de cursor nog een sprong maken. Controleer dat voor uzelf door de waarde in geheugenlokatie 201 te veranderen in 4 en het programma nogmaals te runnen.

Andere schermen

Tot nu toe heeft u alleen voorbeelden gezien in het normale tekstscherm van de Atari, zoals gezegd 24 regels van 40 lettertekens breed.
De Atari kent echter nog **vier** andere tekstschermen die we hier in het kort willen aanduiden. De opdracht:

GRAPHICS nr.

zorgt voor de keuze van een bepaald beeldscherm. Voor nr. kunt u een getal invullen van 0 tot en met 15. 0 zorgt voor een gewoon tekstscherm. Getal 1 zorgt voor een beeldscherm van 20 regels van 20 lettertekens. De lettertekens worden extra groot weergegeven. Onderaan het scherm bevindt zich een strook van 4 regels van 40 lettertekens waar de lettertekens normaal worden weergegeven. GRAPHICS 2 geeft hetzelfde effect als scherm 1, maar nu slechts 10 regels van 20 lettertekens. De schermen 12 en 13 zorgen voor samengedrukte teksten, respectievelijk 20 en 10 regels van 40 lettertekens.
Om onderscheid te kunnen maken tussen het grote tekstscherm en de strook onderaan heeft u een extra aanduiding bij de PRINT opdracht nodig. Voegt u aan het vorige programma een regel toe:

5 GRAPHICS 1

en run het programma. U ziet nu een zwart scherm met onderin een blauwe balk met daarin de mededeling READY. Het blijkt dat de normale PRINT opdracht de teksten in normaal lettertype afdrukt in het kleine stukje *'normaal'* beeld: het venster onderin het scherm. Om de tekst met extra grote letters op het scherm te krijgen moet u de computer speciaal mededelen dat de tekst niet in het venster geplaatst moet worden. U gebruikt daarvoor de opdracht:

PRINT#6,

Verander het vorige programma zo dat het resultaat als volgt wordt:

```
5 GRAPHICS 1
10 A=4
20 B=A^2
30 PRINT #6,A,A,A,A,A,A,A,A,A,A
40 PRINT #6,"ATARI","MICRO";"COMPUTER"
```

Als u dit programma runt ziet u de letters twee keer zo groot op het scherm. Als u het gewone scherm weer terug wilt krijgen, kunt u op de RESET knop drukken. U raakt het programma daardoor niet kwijt. Probeer het programma ook te runnen met in regel 5 GRAPHICS 2 in plaats van 1. Probeer tevens welke effecten u krijgt als u de tabulatiesprongen verandert door in geheugenadres 201 een andere waarde dan 10 te POKEn.
Merk op dat de kantlijn bij deze andere tekstvensters is verdwenen.

Tabuleren in twee richtingen

De tot nu toe genoemde tabulatie mogelijkheden zorgen er allemaal voor dat u de plaats van de cursor binnen een bepaalde regel kunt beinvloeden. Het is echter ook zeer goed mogelijk de plaats van de cursor over het gehele scherm te beinvloeden. Daarvoor dient de Atari-BASIC opdracht:

POSITION (=positie)

Achter deze opdracht plaatst u twee getallen. Het eerste getal geeft de positie van de cursor op de regel aan, het tweede getal geeft de regel aan.

```
10 GRAPHICS 0
20 POSITION 0,4
30 PRINT "positie nul, regel 4"
```

Tik dit programma in en RUN het. U ziet dat door de POSITION opdracht de kantlijn vervalt.
Het volgende programma laat zien dat u echt overal op het scherm kunt komen.

```
10 GRAPHICS 0
20 POSITION 5,4
30 PRINT "positie 5, regel 4"
40 POSITION 15,20
50 PRINT "positie 15, regel 20"
```

```
60 POSITION 37,8
70 PRINT "positie 37, regel 8"
```

Merk op dat als u dit programma RUNt, de READY mededeling direkt onder de laatste afgedrukte tekst komt te staan. Dit is volgens regel 70 van het programma de tekst: positie 37, regel 8. Daardoor komt de READY mededeling midden op het scherm te staan, tussen de andere teksten in. Ook de teksten onder de READY mededeling worden niet beinvloed!
Het volgende programma maakt gebruik van de POSITION opdracht om u goed te laten wennen aan de opmaak van het beeldscherm. Tik het programma in en oefen er veel mee.

```
5 GRAPHICS 0
10 POSITION 0,0
20 PRINT "0123456789012345678901234567890123456789"
30 FOR A=0 TO 22
40 POSITION 0,A
50 PRINT A
60 NEXT A
70 POSITION 20,21
80 PRINT "KOLOM ";:INPUT X
90 POSITION 20,22
100 PRINT "REGEL ";:INPUT Y
110 POSITION X,Y
120 PRINT "*(";X;",";Y;")"
130 POSITION 27,21
140 PRINT "   "
150 POSITION 20,22
160 PRINT "          "
170 GOTO 70
```

U kunt het programma onderbreken door de RESET knop in te drukken. Let in het programma op het gebruik van de INPUT opdracht en de verschillende punt-komma's. Ziet u dat er nu wel **40** letterteken posities op het scherm zijn?

Hengelen

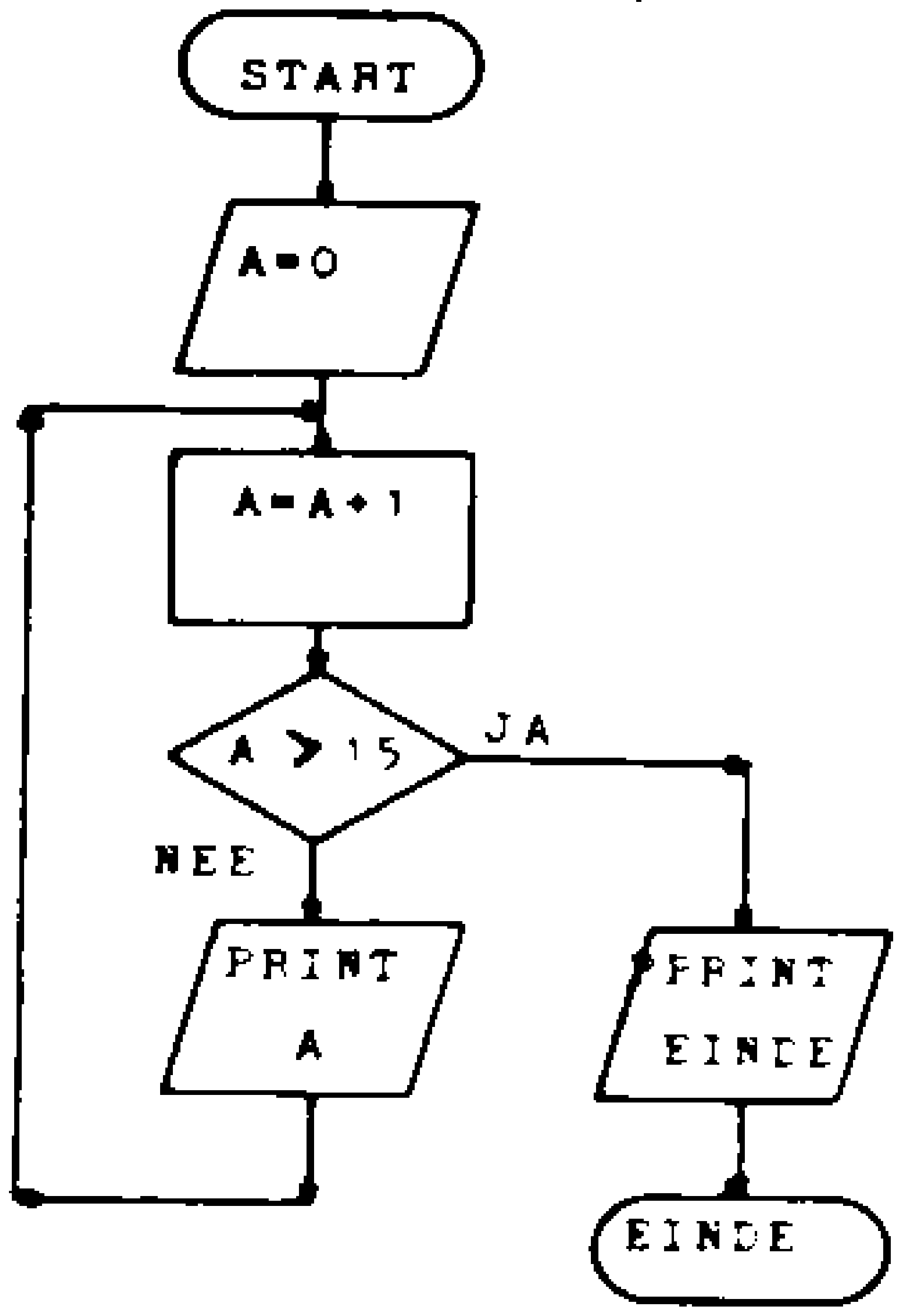
Tot slot nog een leuk programma waarin veel gebruik gemaakt wordt van de POSITION opdracht. Kijk of u het programma begrijpt door alleen de listing te lezen. Maak voor uzelf een stroomdiagram. Er zitten wel veel opdrachten in die tot nu toe nog niet in dit boek behandeld zijn. Mocht het programma u in eerste instantie nog niet duidelijk zijn, lees dan dit boek gewoon verder door. Na verloop van tijd zult u alle hierin gebruikte opdrachten begrijpen en kunt u dit programma er nog eens op nalezen.

```
10 REM hengelen
15 GRAPHICS 0
20 LET S=0
30 LET PU=0
40 PRINT "moeilijkheidsgraad (30-5)"
50 PRINT "30 = gemakkelijk"
60 PRINT "5  = moeilijk";:INPUT M
70 PRINT :PRINT "snelheid (1-5)";:INPUT V
90 GRAPHICS 0
100 POKE 752,1
110 FOR ST=1 TO 20
120 POSITION 25,21:PRINT "score: ";PU;
130 POSITION M+2,0:PRINT "<<+>>";
140 LET RA=INT(RND(0)*20)+1
150 FOR X=1 TO 36
160 POSITION X,RA:PRINT " * ";
170 POSITION M+4,S:PRINT ":";
180 POSITION M+4,S+1:PRINT " ";
190 T=PEEK(764)
200 IF S=0 THEN GOTO 230
210 IF T=6 THEN S=S-1
220 IF S=21 THEN GOTO 240
230 IF T=38 THEN S=S+1
240 POSITION M+4,S:PRINT "#"
250 IF  NOT (M+4=X AND S=RA) THEN GOTO 260
252 SOUND 1,100,10,15
254 FOR WA=1 TO 30:NEXT WA
256 SOUND 1,0,0,0
258 PU=PU+1:GOTO 280
260 FOR Z=1 TO 50-V*10:NEXT Z
270 NEXT X
280 NEXT ST
290 PRINT "EINDE"
300 END
```

RUN het programma. U moet kiezen voor een moeilijkheidsgraad en een snelheid. Boven in beeld verschijnt een *'ruimteschip'*. Van links naar rechts zweven sterren over het scherm. U kunt die vangen met behulp van een 'hengel' die u vanuit het schip kunt laten zakken. De toets met / zorgt ervoor dat de hengel zich naar beneden gaat bewegen. De toets met + zorgt ervoor dat de hengel naar boven gaat bewegen. De spatiebalk (of elke andere toets) zorgt ervoor dat de hengel niet verder beweegt. De POKE opdracht in regel 100 plaatst een waarde groter dan 0 in geheugenlokatie 752. Daardoor wordt de cursor uitgeschakeld. Probeert u hetzelfde programma maar eens met een 0 gePOKEd op adres 752. Het beeld is dan veel onrustiger.

7. LUSSEN

In bijna elk programma is het wel eens handig om de computer een bepaalde handeling een aantal keren te laten verrichten, telkens met een iets ander gegeven, of telkens met dezelfde gegevens. Stel dat het volgende probleem door de computer opgelost moet worden. De getallen van 1 tot en met 15 moeten op het beeldscherm weergegeven worden zonder dat alle vijftien getallen ingetikt hoeven te worden. We kunnen daarvoor een stroomdiagram maken:



PRINT de getallen van 1 tot en met 15

Op deze wijze is er een **loop** (=lus) in het programma gemaakt. De computer draait in het programma een rondje en komt daar pas uit als A groter is dan 15. De eerste keer dat de computer de lus doorloopt, is de waarde van A gelijk aan 0+1=1. De computer drukt dit af, gaat naar de volgende opdracht waar gekeken moet worden of A groter is dan 15. 1 is kleiner dan 15 dus het antwoord is nee. De computer wordt teruggestuurd, telt weer 1 op bij A. A wordt daardoor 2. Dit wordt weer afgedrukt, vergeleken met 15, enz. Dit rondje, (lus) draait de computer net zo lang tot het antwoord op de vraag 'is A groter dan 15?', positief beantwoord kan worden. Als A=16, is A groter dan 15 en gaat de computer niet terug in het programma, maar gaat hij verder met de volgende regel die het einde van het programma aangeeft. Hiermee is een voorbeeld gegeven van een programma waar de computer bij het doorlopen van het programma wel op de volgorde van de regelnummers let, maar waarbij de programmeur ervoor kan zorgen dat de computer bepaalde stukken programma verschillende keren doorloopt. Het programma dat bij bovenstaand stroomdiagram hoort is:

```
10 LET A=0
20 LET A=A+1
30 IF A>15 THEN GOTO 60
40 PRINT A
50 GOTO 20
60 PRINT "EINDE"
```

Van een Atari computer kunt u hier een slimmigheidje voor verwachten. Dat is er dan ook. Om de hierboven genoemde gang van zaken te bespoedigen gebruikt u

FOR...NEXT... (vanaf... volgende...)

Tik NEW en het volgende programma in:

```
10 FOR A=1 TO 15
20 PRINT A
30 NEXT A
40 PRINT "EINDE"
```

Met minder en kortere programmaregels bereikt u hetzelfde resultaat. De opdracht werkt als volgt:
In de eerste regel staat

FOR A = 1 TO 15

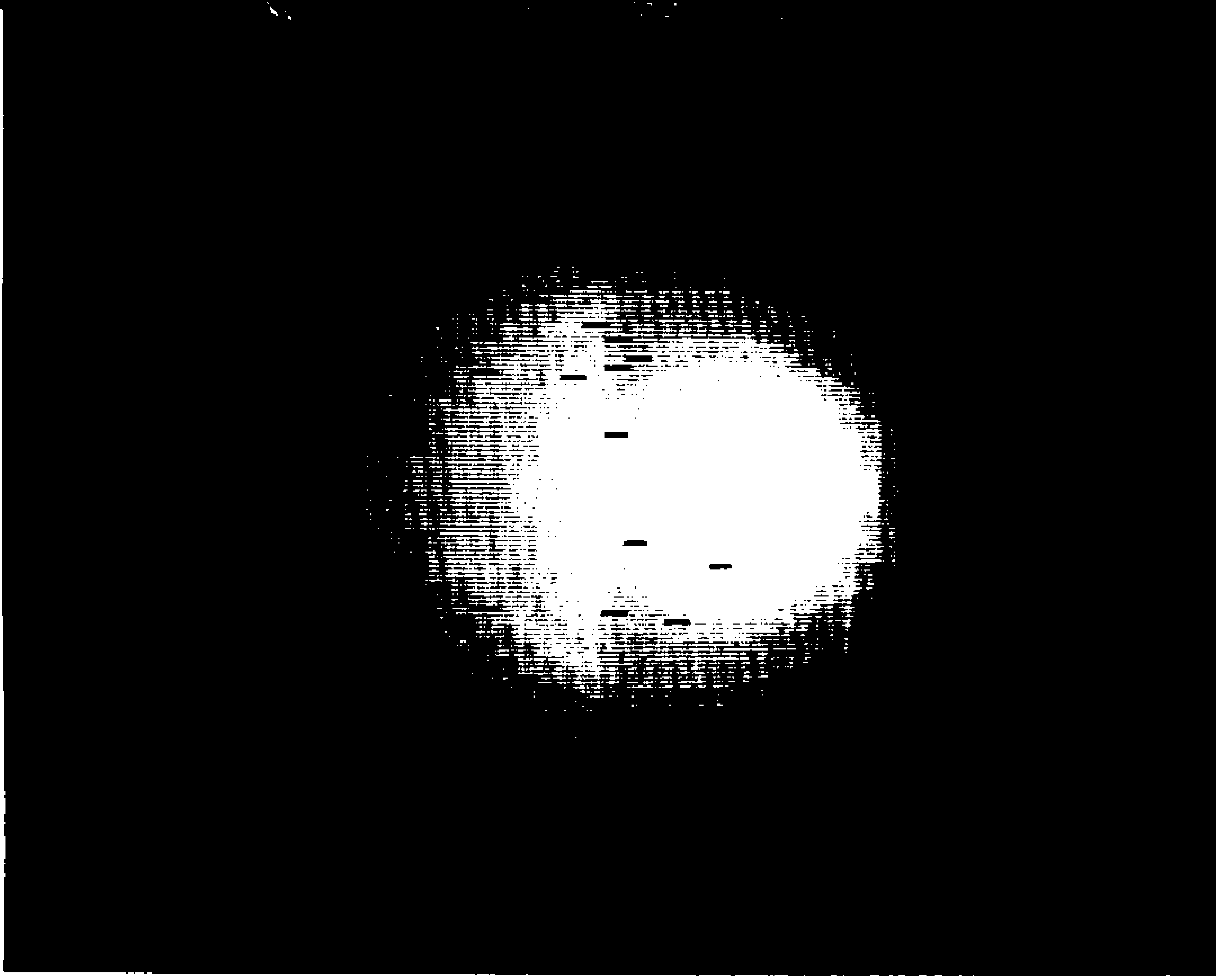
Vertaald in mensentaal wil dat zeggen: computer, de waarde van A gaat van 1 tot en met 15 lopen en zal telkens met 1 worden verhoogd. Begin maar met de eerste waarde.
Regel 20:

PRINT A

Deze regel geeft de computer de bewerking die A moet ondergaan. In dit geval alleen PRINT, maar alle andere BASIC bewerkingen zijn ook mogelijk. Ook BASIC-opdrachten waarbij A helemaal geen rol speelt. Er is geen maximum aantal opdrachten, maar ze moeten allemaal tussen de regel met FOR en de regel met NEXT staan.
Regel 30:

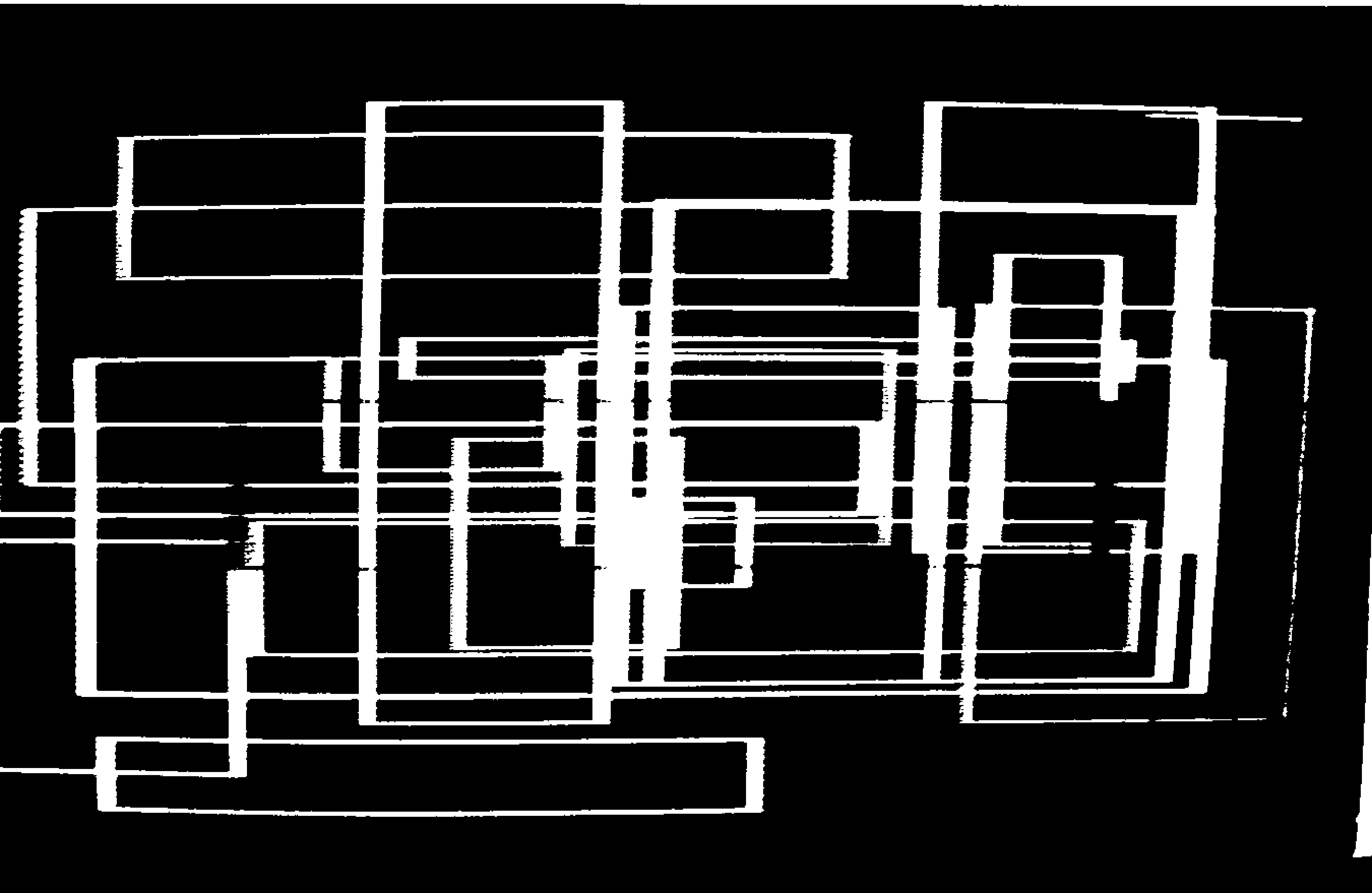
NEXT A

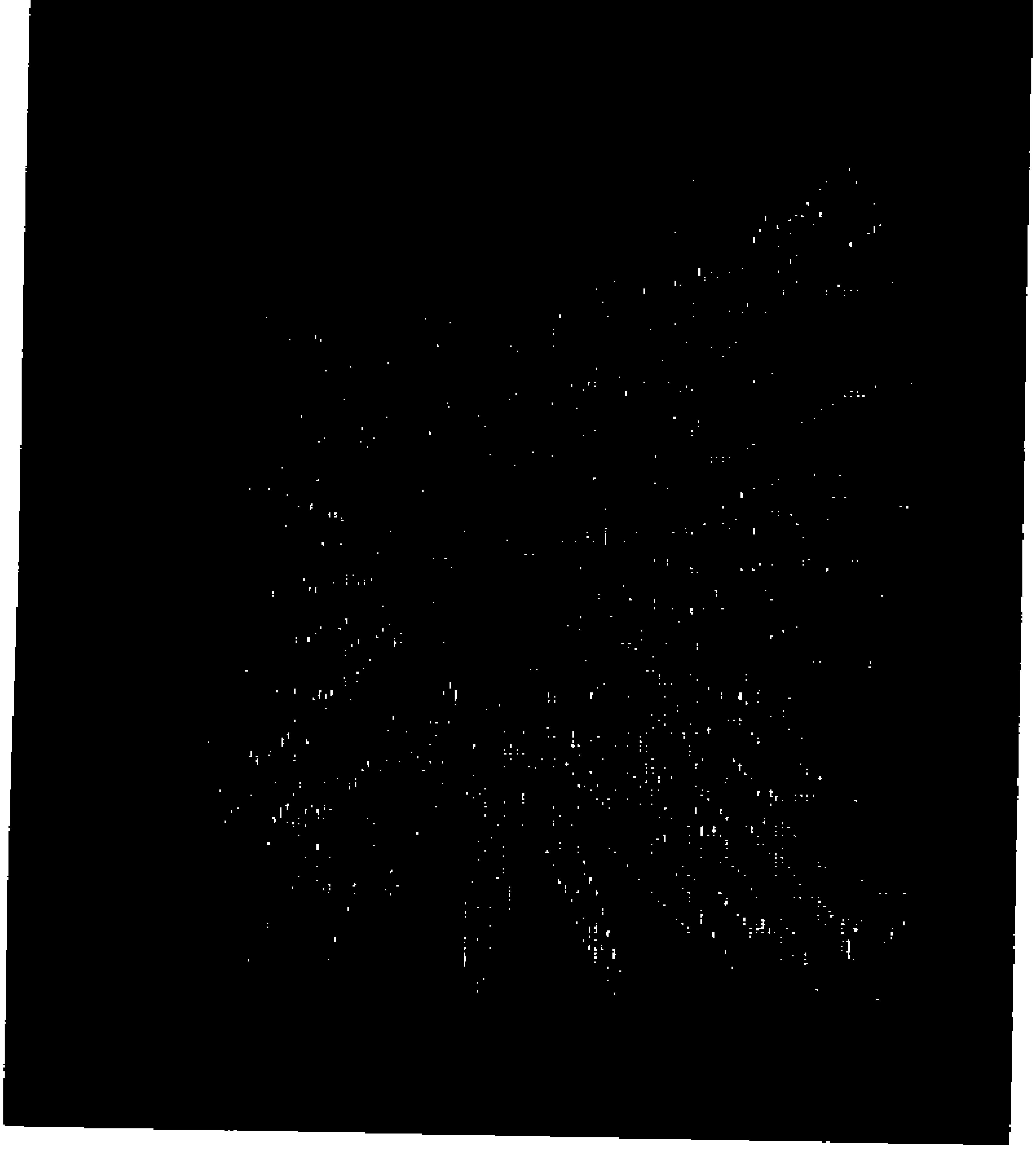
Deze regel vertelt de computer de volgende waarde voor A te nemen, terug te gaan naar de regel waarop FOR staat, daar te kijken of de waarde niet groter is geworden dan het aangegeven bereik (TO 15). Als dat niet zo is, gaat de computer naar de volgende regel en voert daar de bewerking uit met eventueel



Golfhol programma *blz. 250*

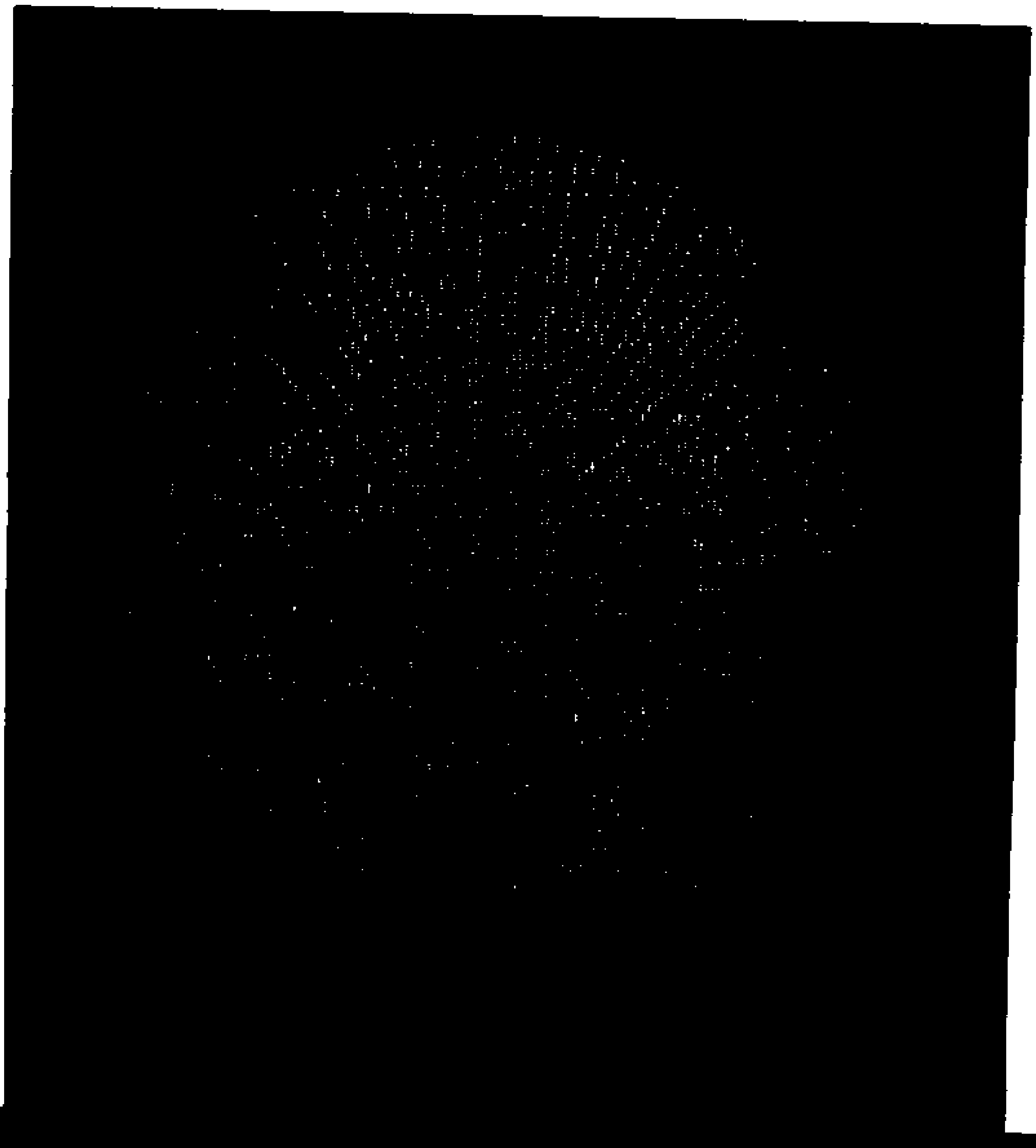
Rechthoeken programma *blz. 55*





Verzameling lijnen programma *hoofdstuk 21*

Graphics voor soortgelijke programma's *hoofdstuk 21*



de nieuwe waarde voor A. Vervolgens komt de computer weer bij regel 30 waar A weer met 1 verhoogd wordt, enzovoort.
Zodra A de waarde 16 (hetgeen groter is dan is) bereikt voert de computer de instruktie uit regel 20 niet meer uit en springt hij naar de regel volgend op de NEXT-statement (statement = formulering van een opdracht). In dit programma betekent dat het einde van het programma.
Met de FOR...NEXT-opdracht kunt u de computer een bepaalde opdracht verschillende keren laten uitvoeren, waarbij u zelf het aantal keren aangeeft en de computer de waarde van een opgegeven variabele verandert. Maar er is meer te doen met de FOR...NEXT-opdracht. De stappen die de computer neemt bij het veranderen van de waarde kunnen we beinvloeden. Tikt u het volgende demonstratieprogramma maar in:

```
10 REM demonstratie FOR...NEXT
20 GRAPHICS 0
30 PRINT "FOR X=16 TO 25 geeft:"
40 PRINT
50 FOR X=16 TO 25
60 PRINT X
70 NEXT X
80 PRINT
90 PRINT
100 REM pauze
110 FOR WA=1 TO 3000:NEXT WA
120 REM ook grotere stappen mogelijk
130 PRINT "FOR X=1 TO 21 STEP 2 geeft:"
140 PRINT
150 FOR X=1 TO 21 STEP 2
160 PRINT X
170 NEXT X
180 PRINT :PRINT
190 REM pauze
200 FOR WA=1 TO 3000:NEXT WA
210 REM omlaag
220 PRINT "FOR X=512 TO 403 STEP -9 geeft:"
230 PRINT
240 FOR X=512 TO 403 STEP -9
250 PRINT X
260 NEXT X
270 PRINT :PRINT
280 REM pauze
290 FOR WA=1 TO 3000:NEXT WA
300 REM ook niet gehele getallen
310 PRINT "FOR X=113.4 TO 3 STEP -18.7 geeft:"
320 PRINT
330 FOR X=113.4 TO 3 STEP -18.7
340 PRINT X
```

```
350 NEXT X
360 PRINT :PRINT
370 REM pauze
380 FOR WA=1 TO 3000:NEXT WA
390 REM variabelen gebruiken
400 PRINT "A=7, B=87, C=8"
410 PRINT "en FOR X=A TO B STEP C geeft: "
420 PRINT
430 LET A=7:LET B=87:LET C=8
440 FOR X=A TO B STEP C
450 PRINT X
460 NEXT X
```

RUN dit programma en u krijgt op het beeldscherm een keur van voorbeelden van mogelijke variaties met behulp van de FOR..NEXT-lus.
Daarnaast ziet u in de listing nog enkele andere belangrijke zaken. Er is al eerder op gewezen, weet u nog; let u op het gebruik van de REM-statement, alles wat daarachter staat wordt door de computer overgeslagen. U kunt uw listing ermee verduidelijken.
Regel 100 geeft als kommentaar alleen *'pauze'*. In regel 110 staat een manier om de FOR...NEXT-lus als *vertrager* te laten werken. De enige opdracht die u opgeeft is om een heleboel keer de waarde van de variabele WA te verhogen. Daar is de computer even mee bezig, en daardoor vertraagt het programma zoveel dat u de resultaten rustig kunt doorlezen. Deze manier van vertragen is echter niet geschikt voor precieze tijdsbepalingen. Om een nauwkeuriger tijdsmeting uit te leggen wijken we even af van de FOR...NEXT-lussen.

Tijd meten

Met de FOR.. NEXT.. lus zijn wel tijden te meten, maar erg precies is het niet. Als vuistregel kunt u aanhouden dat het tellen van 1 tot en met 250 ongeveer één seconde kost.
Atari-BASIC kent geen ingebouwde klok en een speciale opdracht voor het maken van pauzes ontbreekt ook. Bij bovenstaande programma was dat natuurlijk niet zo erg. De pauzes in het programma dienden alleen om de gebruiker de gelegenheid te geven de verschillende mogelijkheden te lezen. Maar als u bijvoorbeeld een klok wilt maken van uw Atari, dan is het belangrijk dat u het tijdsverloop tamelijk nauwkeurig kan meten.
De Atari heeft ergens in het geheugen een zogenaamde beeldteller ingebouwd. Een televisie schrijft 50 beelden per seconde op het scherm. Dit getal 50 komt overeen met de frequentie van het lichtnet die 50 Hertz bedraagt. Omdat de computer stuursignalen moet geven aan de televisie, stuurt de computer gegevens naar de televisie waaruit afgeleid kan worden wanneer een nieuw beeld opgebouwd moet worden. Het aantal op het scherm geschreven beelden wordt bijgehouden in drie verschillende geheugenadressen.
Vanaf het moment dat de computer aanstaat worden deze adressen elke één

vijftigste van een seconde met 1 verhoogd. Ze beginnen alledrie met 0. Omdat de Atari een 8-bits computer is, kan het maximale getal in een geheugenadres gelijk aan 255 zijn (zie ook *bijlage 9*). De beeldtellers zijn echter zoals kilometertellers van een auto: het is mogelijk 'de klok rond te lopen'. De eerste teller loopt van 0 tot en met 255 en begint dan weer bij 0. Als de eerste teller van 255 naar 0 springt, verspringt de tweede teller eentje verder. Dat wil zeggen dat de tweede teller 256 keer langzamer loopt dan de eerste. De derde teller op zijn beurt verspringt pas als de tweede teller van 255 naar 0 springt. Dat wil zeggen nadat de eerste teller 256*256 keren verhoogd is. In totaal kunnen de drie tellers 256 * 256 * 256 tellen bijhouden. Elke een vijftigste van een seconde wordt er geteld. Voordat alle drie de tellers 'volgeteld' zijn, moet uw computer meer dan 93 uur aanstaan!
Met de wetenschap van dit tellen kunnen we de tijd gaan meten. Daarvoor moeten we wel de inhoud van de drie geheugenadressen, die samen de beeldteller vormen,weten. Daarvoor maken we gebruik van de opdracht

PEEK (= spiek)

Deze opdracht kijkt welke waarde er in een bepaalde geheugenlokatie staat zonder iets aan die waarde te doen. Het is de tegengestelde opdracht van de POKE-opdracht die u al eerder tegenkwam.
De beeldteller bevindt zich in de geheugenlokaties 20 (de snelste teller), 19 (de middelste) en 18 (de langzaamste).
De tijd kan afgelezen worden door:

(256*256*PEEK(18) + 256*PEEK(19) + PEEK(20)) / 50

Tikt u PRINT gevolgd door deze regel in en druk vervolgens op RETURN. U krijgt dan de tijd dat uw Atari aanstaat in seconden.
Om de tijd in een programma te meten, moeten de waarden van de drie geheugenadressen wel eerst gelijk aan 0 gemaakt worden. Het is in een programma meestal de bedoeling een bepaald tijdsverloop te meten en niet de tijd te weten die de computer al aanstaat. Het op nul stellen van de beeldtellers gaat met behulp van de eerder genoemde POKE opdracht:

POKE 18,0
POKE 19,0
POKE 20,0

Denkt u eraan dat de snelst tellende teller het laatste op nul gezet moet worden, anders heeft die alweer een paar tellen gedaan tegen de tijd dat de andere tellers op 0 gezet zijn.
Om in een programma een bepaalde en precieze hoeveelheid tijd uit te trekken kunt u het volgende stukje programma gebruiken:

```
10 POKE 18,0:POKE 19,0:POKE 20,0
20 PRINT "WE WACHTEN..........."
30 IF (256*256*PEEK(18)+256*PEEK(19)+PEEK(20))/50<10 THEN GOTO 30
40 PRINT "10 SECONDEN"
```

In regel 10 worden de beeldtellers op 0 gezet. In regel 20 wordt een bericht afgedrukt.
Regel 30 vormt een lus die de hele tijd herhaald wordt, totdat de beeldtellers 500 tellen, dus 10 seconden lang hebben afgeteld.
Om langer dan een minuut te tellen kunt u de volgende formules toepassen:

TIJD = 256*256*PEEK(18)+256*PEEK(19)+PEEK(20)
M = TIJD/3000
S = TIJD/50 - M*60

Er gaan 50 een vijftigste van een seconde in een seconde, maar na 60 seconden moet de klok weer bij 1 beginnen. Daarom wordt telkens 60 maal het aantal minuten van het totale aantal seconden afgetrokken.
Ook uren kunnen in minuten worden uitgedrukt. Een digitale klok kunt u als volgt maken:

```
10 GRAPHICS 1
20 POKE 18,0:POKE 19,0:POKE 20,0
30 U=0
40 TIJD=256*256*PEEK(18)+256*PEEK(19)+PEEK(20)
50 M=INT(TIJD/3000)
60 S=INT(TIJD/50-M*60)
70 IF M>0 AND INT(M/60)=M/60 THEN U=U+1
80 POSITION 4,8
90 PRINT #6;"***************"
100 POSITION 6,10:PRINT #6;"uur ";U
110 POSITION 6,11:PRINT #6;"min ";M
120 POSITION 6,12:PRINT #6;"sec ";S
130 POSITION 4,14:PRINT #6;"***************"
140 GOTO 40
```

Terug naar de lussen

De FOR.. NEXT.. lus kan ook voor andere doeleinden gebruikt worden, dan voor het laten veranderen van een variabele. U kunt de opdracht gebruiken om de computer een bepaalde onveranderlijke opdracht een bepaald aantal keren te laten herhalen. De lus dient dan enkel als een teller. Een voorbeeld hiervan heeft u als pauzelus al gezien. Als u graag uw naam op de televisie ziet kunt u tikken:

```
10 FOR A=1 TO 22
20 PRINT ".....(uw naam....."
30 NEXT A
```

Op deze wijze telt de computer wel telkens 1 op bij A, maar gebruikt hij A niet anders dan om de tel bij te houden.
Het aantal opdrachten dat herhaald moet worden bepaalt uzelf. Elke opdracht die tussen de FOR regel en de NEXT regel staat wordt net zo vaak uitgevoerd als de computer de lus doorloopt.
Het volgende programmagrapje maakt gebruik van een lus om telkens dezelfde opdracht te geven, maar tijdens het doorlopen van de lus de waarden waarmee de opdracht wordt uitgevoerd telkens een beetje te veranderen.

```
10 Q=10:R=60
20 S=1:T=S/1.6
25 PRINT "0< FACTOR <360";:INPUT VD
30 GRAPHICS 11
40 V=90:V0=3.14/180
50 X0=40:Y0=100
60 FOR Z=1 TO 100
70 X=X0+Q*COS(V*V0)
80 Y=Y0+R*SIN(V*V0)
90 IF X<0 OR X>79 OR Y<0 OR Y>192 THEN GOTO 90
95 COLOR Z
100 PLOT X0,Y0:DRAWTO X,Y
110 X0=X:Y0=Y
120 V=V+VD
130 Q=Q+S
140 R=R+T
150 NEXT Z
160 GOTO 160
```

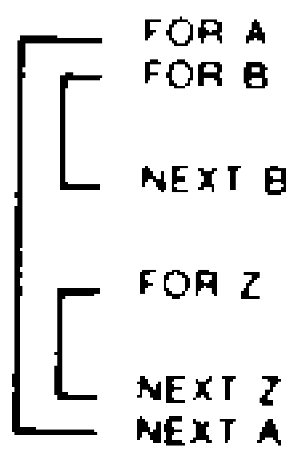
De lus loopt van regel 60 tot en met regel 150. De opdracht die zichtbaar op het scherm komt zit in regel 100. De computer moet tussen twee punten een lijn trekken. Maar in dezelfde lus waarin de computer telkens de lijn moet trekken worden de waarden van de punten waartussen de lijn getrokken moet worden veranderd.
Tik het programma in en RUN het. U moet nu van de computer een factor invoeren. Deze factor bepaalt hoe de tekening op het scherm er uit komt te zien. Begin met 182 in te vullen. Langzaam verschijnt een tekening op het scherm. Duidelijk is te zien hoe de computer steeds een lijn tussen twee punten trekt, maar elke keer dat hij door de lus loopt, de punten een beetje verandert. Regel 90 zorgt ervoor dat de computer geen foutmelding geeft als de punten buiten het beeld dreigen te vallen. In dat geval wordt het programma in een oneindige lus geplaatst.
Regel 160 zorgt na de tekening ook voor een oneindige lus. Zonder deze lus zou de tekening gelijk weer verdwijnen.
Leuke waarden om als factor in te vullen zijn 70, 91, 120, 150, 225, 230, 270, 270.5, 300

Lussen binnen lussen

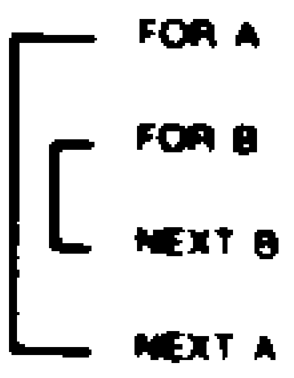
U kunt ook lussen binnen lussen maken. Deze worden *genestelde* lussen genoemd. De lussen liggen als nesten binnen elkaar. De binnenste lus wordt elke keer als de buitenste lus wordt doorlopen net, zo vaak doorlopen als u aangeeft. Een voorbeeld maakt dit duidelijk. Tik NEW en het volgende programma:

```
10 FOR A=1 TO 3
20 PRINT "BUITENSTE LUS NR. ";A
30 PRINT
40 FOR B=1 TO 4
50 PRINT "binnenste lus nr. ";B
60 NEXT B
70 PRINT
80 FOR Z=1 TO 2000:NEXT Z
90 NEXT A
```

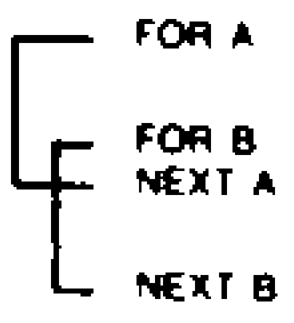
Dit programma heeft drie lussen, waarvan eentje de speciale pauzelus vormt. De lus FOR B=1 TO 4..NEXT B ligt helemaal binnen de lus FOR A=1 TO 3..NEXT A. Naast de B-lus ligt de lus FOR Z=1 TO 2000..NEXT Z. Ook deze lus ligt binnen de A-lus. In schema zien de lussen er als volgt uit:



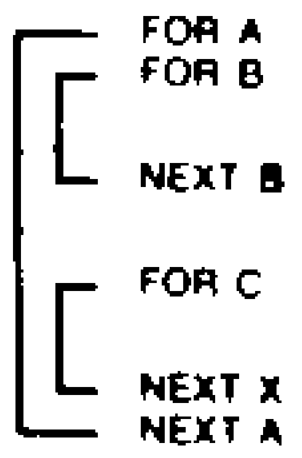
Let u er bij het leggen van genestelde lussen goed op dat de binnenste lus ook echt binnen de buitenste lus ligt. **De lussen mogen elkaar niet kruisen!**



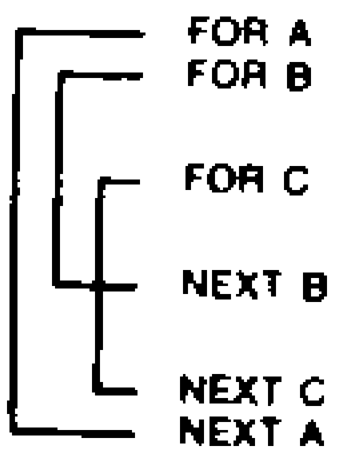
goed



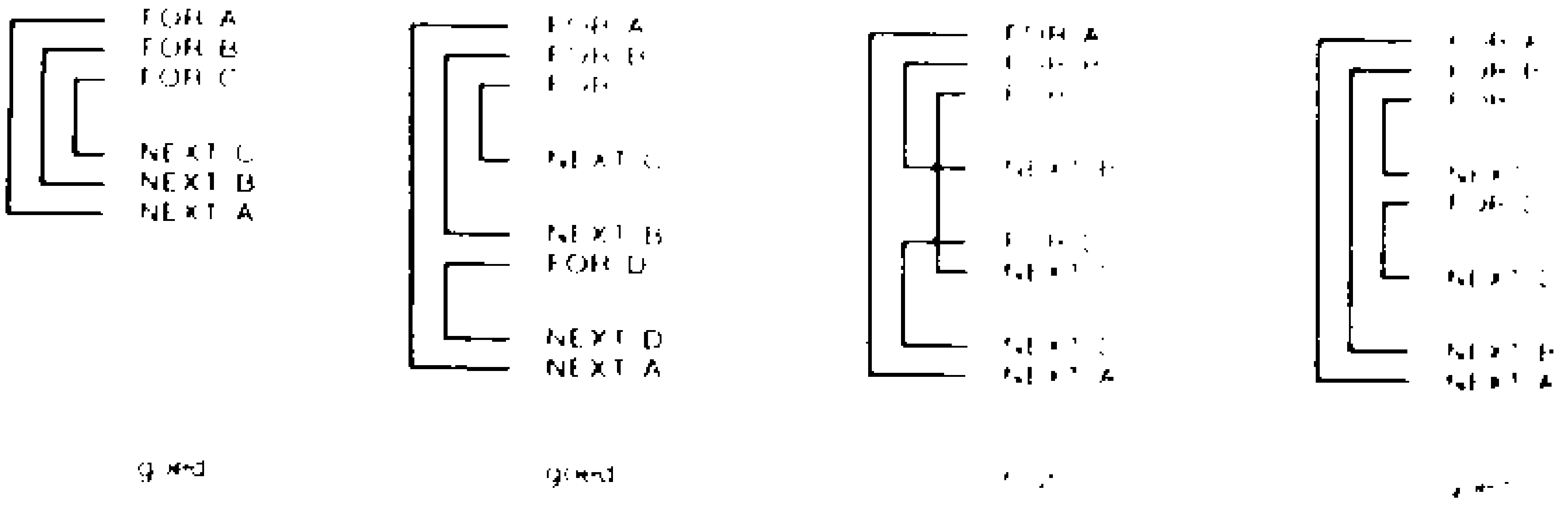
fout



goed



fout



voorbeelden van genestelde lussen

Tot slot van dit hoofdstuk nog twee verschillende korte rekenprogramma's. Het eerste programma rekent voor u het gemiddelde van een stel getallen uit.

```
10 GRAPHICS 0
20 PRINT "Ik reken voor u het gemiddelde uit"
30 PRINT "van een lijst door u opgegeven"
40 PRINT "getallen."
50 PRINT :PRINT "HOEVEEL GETALLEN IN DE LIJST";:INPUT N
60 PRINT :PRINT
70 PRINT "Druk na elk getal op RETURN"
80 T=0
90 FOR L=1 TO N
100 PRINT "getal nr.";L;" = ";:INPUT X
110 T=T+X
120 NEXT L
130 G=T/N
140 PRINT :PRINT "Het totaal = ";T
150 PRINT :PRINT "Het GEMIDDELDE = ";G
```

Het volgende programma berekent de faculteit van een getal. De faculteit van bijvoorbeeld het getal N wordt aangegeven met N! (spreek uit: *en faculteit*). Dit staat voor N*(N-1)*(N-2)*(N-3)*....3*2*1. Normaal gesproken wordt dit alleen gebruikt bij positieve gehele getallen.
Het programma gaat er als volgt uitzien:

```
10 GRAPHICS 0
20 PRINT :PRINT "Berekening van FACULTEIT"
30 PRINT
40 PRINT "EEN GETAL A.U.B.";
50 INPUT G
60 LET A=G
```

```
70 LET B=G
80 FOR A=G-1 TO 1 STEP -1
90 LET B=A*B
100 NEXT A
110 PRINT :PRINT
120 PRINT G;" faculteit is ";B
130 PRINT
140 PRINT "Tik 1, RETURN voor nog een keer."
150 PRINT "Tik 0, RETURN voor stop."
160 INPUT N
170 IF N=1 THEN GOTO 30
180 PRINT "EINDE"
```

Als u dit programma ingetikt en geRUNd heeft, kunt u van allerlei getallen snel de faculteit vinden. Het programma is niet 'beveiligd' tegen onjuist gebruik. Als de gebruiker een negatief getal invoert, werkt het niet goed. Een breuk kan ook niet veilig worden ingevoerd. Wat gebeurt er als u een getal groter dan 68 intikt? Wat betekent die foutmelding? **Ziet u dat de notering van de getallen vanaf 14! anders is.**

8. LOGICA EN IF...THEN...

Vergelijken

Uw Atari is een mooie computer, maar zelfstandig denken is er niet bij. Beslissingen neemt de Atari dan ook niet, maar deze computer kan wel twee getallen vergelijken. Het onderlinge verband tussen twee getallen kan de Atari brengen onder een van de volgende noemers:

=	is gelijk
<>	is niet gelijk
<	kleiner dan
>	groter dan
< =	kleiner of gelijk dan
> =	groter of gelijk dan

Let u op de afwijkingen met de notaties voor deze vergelijkingen zoals u die wellicht kent uit de wiskunde:

<> wordt normaal aangegeven met $\neq$
> = wordt normaal aangegeven met $\geq$
< = wordt normaal aangegeven met $\leq$

Als de computer twee getallen heeft vergeleken, kunnen we aan de hand van de uitkomst, de computer een bewerking wel of niet laten uitvoeren, een waarde aan een variabele laten toekennen of naar een regel in het programma laten springen. De hiervoor gebruikte opdracht is:

IF...THEN... (als...dan...)

De eerste stippeltjes van IF...THEN... staan voor het vergelijken van twee waarden. Als de waarden zich verhouden zoals het er tussen staande teken (uit de lijst hierboven) aangeeft (dat wil zeggen dat de bewering waar is) dan (THEN) voert de computer de opdracht(en) op het tweede stel stippen uit. We kunnen ook zeggen: IF de bewering is waar THEN voer de opdracht uit. In de praktijk blijkt dit een zeer nuttige faciliteit te zijn, want op deze manier kan de computer bepaalde *'beslissingen'* nemen. Hier volgen een paar voorbeelden:

IF X < 10 THEN GOTO 300
Als de waarde van X kleiner is dan 10 of anders gesteld als de bewering X < 10 waar is, gaat de computer naar regel 300 (THEN GOTO 300). Zo niet (X < 10 is niet-waar), dan vervolgt de computer het programma bij de volgende programmaregel, zonder de opdracht achter THEN uit te voeren. In een miniprogramma ziet het er zo uit:

```
10 INPUT X
20 IF X<10 THEN GOTO 300
30 PRINT "X is groter dan of gelijk aan 10"
40 END
300 PRINT "X is kleiner dan 10"
```

Let erop dat een variabele alleen dan als waarde in een vergelijking gebruikt mag worden als de variabele eerder in het programma een waarde toegekend heeft gekregen. *De Atari beschouwt anders de variabele als 0,* hetgeen tot ongewenste effecten kan leiden.

IF A > B THEN GOTO 300
Als de waarde van A groter is dan de waarde van B (A > B is waar) dan gaat de computer naar regel 300. Zo niet, (A > B is niet-waar) dan vervolgt hij het programma met de volgende regel.

IF Y=0 THEN LET Z=6
Als de waarde van Y gelijk is aan 0 (Y=0 is waar) dan kent de computer aan de variabele Z de waarde 6 toe en vervolgt daarna het programma met de volgende regel. Is de bewering niet-waar dan vervolgt de computer het programma met de volgende regel.

IF K < > L THEN PRINT "K is niet gelijk aan L"
De tekst tussen aanhalingstekens wordt alleen afgedrukt als K niet gelijk is aan L,dt wil zeggen K < > L is waar.

IF P > =Q THEN GRAPHICS 0
Het scherm wordt gewist (schermmodus 0 wordt gekozen) als de bewering P > =Q waar is. Anders gaat de computer direkt door met de volgende programmaregel.

IF T < =S THEN SOUND 1,10,50,11
Het programma verzorgt een toeter als T kleiner dan of gelijk aan S is. Zie als voorbeeld dit programma:

```
10 LET S=11
20 FOR T=20 TO 6 STEP -1
30 PRINT "T=";T,"S=";S
40 IF T<=S THEN SOUND 1,5,50,11
50 FOR X=1 TO 600:NEXT X
60 SOUND 1,0,0,0
70 NEXT T
80 END
```

Bij de opdracht IF ..1.. THEN ..2.., kan bij ..2.. elke BASIC opdracht (of een combinatie opdrachten) geschreven worden. Als de bewering ..1.. waar is, voert de computer de opdracht(en) uit. Als de bewering niet waar is negeert de computer de opdrachten bij ..2.. en gaat hij door met de volgende programmaregel. Naast deze algemene BASIC opdracht kennen enkele andere vormen van BASIC nog een extra opdracht die de Atari niet kent. Voor het gemak bij het omzetten van programma's hier een korte aanduiding:
IF...THEN...ELSE... (als...dan...anders...)
Deze opdracht zorgt ervoor dat de computer niet onmiddellijk naar de volgende regel springt als de bewering achter IF niet-waar is, maar eerst de opdracht achter ELSE uitvoert. Bijvoorbeeld:
IF A > Z THEN GOTO 200 ELSE PRINT " "
Als de bewering A > Z waar is gaat de computer naar 200, anders (dat wil zeggen A > Z is niet waar) drukt de computer een spatie af.

De volgende punten bij het gebruik van de IF...THEN... vertakkingen zijn nog te vermelden:
Als achter THEN de opdracht GOTO regelnummer komt, mag GOTO weggelaten worden.

IF X=Z THEN GOTO 200
IF X=Z THEN 200 Deze regels betekenen hetzelfde

Na THEN mag meer dan een opdracht gezet worden. De opdrachten moeten gescheiden zijn door **dubbele punten** (:). De opdrachten achter THEN worden of allemaal uitgevoerd (bewering waar) of geen van alle uitgevoerd (bewering niet waar).

Logische operators

Bij het programmeren in BASIC zult u AND, OR en NOT vooral als logische operators (bewerkers) tegenkomen. Zij werken dan als volgt:

ccc AND ddd	is alleen waar als ccc waar is EN ddd ook waar is.
ccc OR ddd	is alleen waar als ccc waar is OF ddd waar is OF allebei waar zijn.
NOT ccc	is alleen waar als ccc NIET waar is.

Een paar voorbeelden:

IF X > 0 AND X < 1 THEN ...
Als de waarde van X groter is dan 0 en de waarde van X is ook kleiner dan 1, dan is de totale bewering waar en zal de computer de opdracht achter THEN uitvoeren.

IF X > 0 OR Y < 1 THEN ...
Als de waarde van X groter is dan 0 of de waarde van Y kleiner is dan 1, of alle-

bei waar is, dan is de hele bewering waar. De computer voert de opdracht achter THEN uit.

IF 2+3=5 AND 4+7=11 is waar
IF 2+3=5 OR 4+3=11 is ook waar

Dit ziet er vreemd uit: Bij zo'n berekening van 2+3=5 kunnen we zelf toch wel zien of dat waar of niet waar is. Door dit voorbeeld wordt echter duidelijk dat de computer bij de IF..THEN.. opdracht uitsluitend kijkt naar het waar of niet-waar zijn van het gestelde achter IF. In het vorige programma kunt u in regel 40 T <=S vervangen door 2+3=5. Deze bewering is waar. De computer zal telkens de toeter opnieuw laten horen. Probeer ook bovengenoemde combinaties met AND en OR uit.

IF NOT X=Y*A THEN ...
Als de waarde van X niet gelijk is aan Y*A is de bewering NOT X=Y*A waar. De opdracht achter THEN wordt uitgevoerd.

De logische operators kunnen ook in combinatie worden gebruikt:

IF NOT (X=1 AND Y=9) THEN ...
IF (X=1 AND Y=9) OR (X=-1 AND Y=-9) THEN ...
IF (X=1 OR X=9) AND (A=-1 OR A=-9) THEN ...

Bij de eerste regel is de gehele bewering waar als X niet gelijk is aan 1 of Y niet gelijk is aan 9.
Bij de tweede regel is de gehele bewering waar als (X=1 en Y=9) of als (X=-1 en Y=-9). Bij de derde regel is de gehele bewering waar bij de volgende combinaties:

X=1 en A=-1
X=1 en A=-9
X=9 en A=-1
X=9 en A=-9

U ziet dat de logische operators net als de rekenkundige operators soms haakjes nodig hebben. Als in een vergelijking zowel logische als rekenkundige operators worden gebruikt, dan worden eerst de rekenkundige operators uitgewerkt. U kunt deze volgorde veranderen door gebruik te maken van haakjes.

Gebruik van logische operators

Het gebruik van de logische operators dient met de nodige voorzichtigheid te gebeuren. Het gebruik van deze logica leidt snel tot fouten die zeer moeilijk te herkennen zijn, en dus moeilijk uit het programma gehaald kunnen worden.

IF X >0 OR X < 1 THEN...
Dit is een zinloze bewering. Als de waarde van X groter is dan 0 of de waarde van X kleiner is dan 1, of allebei, is de bewering waar. Deze bewering is dus altijd waar. Er bestaat geen waarde voor X die kleiner is dan 0 en groter is dan 1.

IF P >0 AND P < 0 THEN PRINT "het perpetuum mobile"
Er bestaat geen enkele waarde voor P die zowel groter dan 0 als kleiner dan 0 is. Deze bewering is altijd niet-waar en de opdracht achter THEN zal, ongeacht de rest van het programma, nooit uitgevoerd worden.
Uiteraard kent de Atari voor dit soort fouten geen foutmelding. Bij een niet-ware bewering wordt de opdracht na THEN gewoon genegeerd en gaat de computer verder met de volgende programmaregel. De computer kan niet kontroleren of u onzinnige beweringen laat uittesten.
Een zeer eenvoudig programma waarin IF...THEN... gebruikt wordt is het onderstaande getal-raad programma.

```
10 REM RAAD HET GETAL
20 GRAPHICS 1
30 LET X=INT(RND(0)*100)
40 PRINT #6;"  U moet een getal"
45 PRINT #6
50 PRINT #6;"  tussen 1 en 100"
55 PRINT #6
60 PRINT #6;"       raden."
65 PRINT #6
70 PRINT #6;"    raadt u goed, "
75 PRINT #6
80 PRINT #6;"     dan heeft u "
85 PRINT #6
90 PRINT #6;"       GEWONNEN"
100 POSITION 2,18
110 PRINT #6;"WELK GETAL GOKT U"
120 INPUT G
130 IF G=X THEN GOTO 200
140 IF G<X THEN POSITION 2,18:PRINT #6;"     TE LAAG    "
150 IF G>X THEN POSITION 2,18:PRINT #6;"     TE HOOG    "
160 FOR WA=1 TO 400:NEXT WA
170 GOTO 100
200 FOR S=150 TO 1 STEP -10
210 SOUND 1,S,10,10
220 FOR WA=1 TO 30:NEXT WA
230 SOUND 1,0,0,0
240 NEXT S
250 GRAPHICS 2
260 POSITION 2,5
270 PRINT #6;"PrImA"
```

Nieuw in dit programma is de opdracht

RND() (RaNDom = willekeurig)

Deze opdracht zorgt ervoor dat de computer een willekeurig getal voor ons uitzoekt. Tussen de haakjes kunt u elk getal invullen dat u wilt, voor het kiezen van het willekeurige getal maakt dat niet uit. Gebruikelijk is om 0 in te vullen. De totale opdracht wordt dan:

RND(0)

De willekeurige waarde die de computer voor u kiest, ligt tussen 0 en 1. 0 kan gekozen worden, maar 1 komt niet voor.
In regel 30 van het programma wordt X een willekeurige waarde gegeven. Deze opdracht ziet er alleen heel wat uitgebreider uit dan hierboven wordt aangegeven .
Het getal dat de funktie RND levert ligt altijd tussen 0 en 1. In het programma waren getallen van 1 tot en met 100 nodig. Om dit te bereiken moet de waarde van RND met 100 vermenigvuldigd worden.
Om een geheel getal te krijgen is het nodig dat het verkregen getal (RND(0)*100) wordt afgerond. Daarvoor dient de opdracht:

INT() (INTeger = geheel getal)

De integerfunktie vormt een getal om in een geheel getal. Daarvoor wordt het dichtsbijzijnde getal genomen dat kleiner is dan het opgegeven getal. De cijfers achter de komma worden verwaarloosd.

```
PRINT INT(5)      geeft 5
PRINT INT(4.9)    geeft 4
PRINT INT(5.1)    geeft 5
```

Let op! De INT funktie rondt het getal dus niet af, maar geeft het grootste gehele getal dat kleiner is dan het opgegeven getal. Bij negatieve getallen wordt ook het grootste gehele getal dat kleiner is dan het opgegeven getal genomen:

```
PRINT INT(-5)     geeft -5
PRINT INT(-4.9)   geeft -5
PRINT INT(-5.1)   geeft -6
```

Het is niet nodig dat u tussen de haakjes van INT een getal plaatst. Het mag ook een variabele zijn, of zoals in het programma een RND getal.

INT(RND(0)*100)

geeft dus een willekeurige geheel getal tussen 0 en 99 (geen 100, want 1 komt **als RND getal niet** voor).

Het bovenstaande programma is eenvoudig uit te breiden tot een spel dat op mastermind lijkt.
Probeert u zelf een stroomdiagram en een programma te schrijven dat:

- vier willekeurige getallen kiest
- de gebruiker om vier getallen vraagt
- de geraden getallen vergelijkt met de willekeurig gekozen getallen
- de gebruiker punten geeft voor het aantal juiste getallen op de juiste plaats
- het spel beeindigt als alle getallen geraden zijn
- dit alles van voldoende instrukties en beeldweergaven voorziet

Als u er zelf niet uitkomt (eerst proberen!) kunt u hieronder een mogelijke oplossing vinden.
U kunt ook uw resultaat vergelijken. Als u aan dezelfde eisen in een ander, goed werkend programma voldaan heeft, wil dat zeker niet zeggen dat uw programma minder goed is. Er zijn vaak verschillende programma's mogelijk voor een bepaald probleem.

```
10 GRAPHICS 1
20 W=INT(RND(0)*10+1)
30 X=INT(RND(0)*10+1)
40 Y=INT(RND(0)*10+1)
50 Z=INT(RND(0)*10+1)
80 PRINT #6;"RAAD 4 GETALLEN"
90 PRINT #6
100 PRINT #6;"4 getallen svp"
110 P=0
120 INPUT A,B,C,D
130 IF A=W AND B=X AND C=Y AND D=Z THEN GOTO 500
140 IF A=W THEN P=P+1
150 IF B=X THEN P=P+1
160 IF C=Y THEN P=P+1
170 IF D=Z THEN P=P+1
180 POSITION 2,10
190 PRINT #6;A;B;C;D
200 PRINT #6:PRINT #6
210 PRINT #6;P;" goed"
220 FOR WA=1 TO 1200:NEXT WA
230 GRAPHICS 1
270 GOTO 90
280 END
500 GRAPHICS 2
510 POSITION 1,7
520 PRINT #6;A;B;C;D
530 PRINT #6
540 PRINT #6;"  PrImA"
550 END
```

Probeert u uit deze programmalisting te halen hoe u met het grote letterscherm kunt werken. Let daarbij op het gebruik van #6 bij elke PRINT opdracht. Gele letters krijgt u door in de te PRINTen tekst kleine letters te gebruiken. Rode letters krijgt u door in de te PRINTen tekst hoofdletters te gebruiken. Zie bijvoorbeeld het woord *'PrImA'*.

9. STRINGS: de computer als tekstverwerker

Wat is een string?

In dit hoofdstuk gaan we de mogelijkheden van de computer vergroten. Tot nu toe zijn alleen nog maar cijfers en getallen verwerkt en is tekst gebruikt om iets weer te geven op het scherm. Nu bekijken we hoe tekst bewerkt en verwerkt kan worden door de Atari.
Om tekst te kunnen verwerken gebruikt de Atari zogenaamde **STRINGS** (=slierten, vrij vertaald: rijtjes). Een string is een aaneengeschakelde rij van karakters. Elk karakter kan gebruikt worden en de string kan net zo *lang worden als u wilt*, tot het geheugen van uw computer vol is. Test zelf uit hoeveel geheugenruimte uw computer heeft door:

PRINT FRE(0)

U krijgt nu het aantal bytes (geheugenplaats waar een letterteken in past) dat nog vrij beschikbaar is. Ook als u een programma aan het schrijven bent, kunt u tussentijds kijken hoeveel geheugenruimte u nog heeft. Gebruikers van een Atari 130 XE worden voor de extra geheugenruimte verwezen naar *bijlage 7*.

U moet er altijd aan denken dat u de computer meedeelt dat u hem wilt laten werken met strings. De computer mag een karakter (bijvoorbeeld a of X of %) niet als variabele beschouwen en ook niet (bijvoorbeeld 1 of 6 of 34627.8) als getal. Daarom moet u de hele string altijd tussen aanhalingstekens (") zetten en de string een naam geven gevolgd door een **dollarteken ($)**.

```
LET A$ = "string"
```

Dit wil zeggen: beschouw de karakters s-t-r-i-n-g als een string met de naam A$.
U herkent dit vast al. Waar heeft u dit eerder gebruikt? Inderdaad, bij de PRINT-opdracht. Daar krijgt de string geen naam en hoeft de computer de string alleen maar af te drukken. Maar ook bij de PRINT-opdracht moet de computer de karakters niet als variabelen of getallen beschouwen en ze niet bewerken. Daarom staan de af te drukken karakters tussen aanhalingstekens.
We gaan een stapje verder. Het is mogelijk om de Atari een serie ingetikte karakters als string te laten verwerken. Daarvoor gebruikt u dezelfde INPUT opdracht als bij de invoer van getallen. Alleen vermeldt u nu dat u een string invoert en zorgt u ervoor dat de string opgeslagen wordt in een variabele die een string kan bevatten ($).

```
PRINT "tik een tekstje in";
INPUT A$
```

Deze twee direkte opdrachten zullen echter bij de Atari tot een foutmelding

leiden: ***Atari-BASIC vereist dat u de lengte van een string altijd eerst opgeeft.*** Als u niet eerst de lengte van een string opgeeft, kan de Atari er niet mee werken. Het opgeven van de lengte van een array gebeurt met de opdracht

DIM (DIMensioneer)

Deze opdracht wordt zowel voor het reserveren van stringruimte als voor het reserveren van geheugenruimte voor arrays gebruikt. Op dit laatste komen we nog terug.
De vorm waarin een string 'gedimensioneerd' wordt is:

DIM A$(xx)

Nu kan de stringvariabele A$ een lengte krijgen van xx lettertekens. Dimensioneer A$ nu tot bijvoorbeeld 100 lettertekens : Dim A$ (100).
Tik de opdrachten nogmaals in, wacht op de mededeling op het scherm, tik iets in en druk op RETURN. De computer beschouwt de INPUT nu als een string. U heeft immers door het dollarteken achter de variabele A, te kennen gegeven dat wat er ingetikt wordt als string moet worden gelezen.
We gebruiken de tot nu toe opgedane kennis voor een tamelijk lang maar eenvoudig programma. Leest u het eerst een keer door. Als u het niet helemaal begrijpt kunt u het beter eerst intikken en laten runnen om te zien wat het programma doet. Als u het helemaal begrijpt kunt u de regels naar eigen behoefte veranderen. U heeft met dit programma het allereerste begin van een tekstverwerker op de Atari.

```
10 REM standaardbrief
20 DIM N$(20),A$(25),P$(20),V$(10),T$(12)
30 PRINT "NAAM van jarige: ";:INPUT N$
40 PRINT "ADRES:           ";:INPUT A$
50 PRINT "PLAATS:          ";:INPUT P$
60 PRINT "VERJAARDAGSDATUM:";:INPUT V$
70 PRINT "TELEFOONNR:      ";:INPUT T$
80 GRAPHICS 0
90 PRINT
100 PRINT N$
110 PRINT A$
120 PRINT P$
130 PRINT :PRINT
140 PRINT "Beste ";N$;","
150 PRINT
160 PRINT "Op deze heugelijke dag, ";V$
170 PRINT "feliciteer ik jou,";N$
180 PRINT "hartelijke met je verjaardag."
190 PRINT "Ik hoop dat je vandaag, ";V$
200 PRINT "een leuke dag zult hebben."
210 PRINT "Hoewel ik het erg druk heb, probeer"
```

```
220 PRINT "ik op de ";A$;" in"
230 PRINT P$;" langs te komen."
240 PRINT "Lukt dat niet,dan bel ik ";T$
250 PRINT "om je per telefoon te feliciteren."
260 PRINT "Tot dan, ";N$
270 PRINT "Hartelijke groeten,"
280 PRINT :PRINT
290 PRINT "  XXXXXXXXXXXXXX"
300 GOTO 300
```

Vul bij de XX-en uw eigen naam in en u heeft een persoonlijke verjaardagsbrief aan een van uw vrienden. Als u een printer heeft kunt u het zelfs op papier laten zetten en zodoende verzenden. Maar dit is niet alleen een persoonlijke brief aan die vriend(in), maar door het programma nog een keer te runnen kunt u een andere naam met bijbehorend adres en verjaardagsdatum intikken en zodoende heeft u voor al uw vrienden een persoonlijke verjaardagsbrief. Uiteraard blijft het nu beperkt tot een grapje op uw beeldscherm maar stel u eens voor dat u de beschikking heeft over zo'n programma met een keurige tekst, een computer met een luxe printer waarvan het schrift niet te onderscheiden is van dat van een gewone schrijfmachine en de mogelijkheid om de namen en de adressen door de computer in te laten vullen. Ziedaar de manier waarop al die nette, vriendelijke en hoogst 'persoonlijke' brieven van *boekenverkopers, cursusaanbieders en reclamebureaus* tot stand komen.
Voor echte tekstverwerking komt wel wat meer kijken. Een tekstverwerkend programma moet alle ingetikte tekst kunnen bewerken en verwerken en opslaan. Tot de mogelijkheden van zo'n programma horen:

- correctie van tekst op een willekeurige plaats
- opzoeken van bepaalde woorden of karaktercombinaties
- lay-out van een bladzijde
- tellen van het aantal ingevoerde tekens of woorden.

Voor de Atari computer is op het ogenblik als belangrijkste tekstverwerkend programma **ATARI-WRITER** beschikbaar. Deze tekstverwerker wordt geleverd in de vorm van een ROM-cartridge. Het kost geen extra geheugenruimte om deze cartridge te gebruiken. Daardoor is ATARI-WRITER een snelle tekstverwerker waar redelijk grote teksten in aangemaakt en verwerkt kunnen worden. Op het moment dat dit boek geschreven wordt, is men bezig een nieuwe editie van ATARI-WRITER te maken die het gehele geheugen van de **130XE** kan gebruiken. Hierdoor zal het mogelijk worden nog grotere teksten te verwerken.

Stringvariabelen

Voordat we verder gaan met de strings halen we eerst een stukje over de variabelen op. In het hoofdstuk 'Variabelen' leerde u rekenen met letters in

plaats van cijfers. Zo'n letter gaf aan dat er op die plaats verschillende waarden ingevuld konden worden.
Daarnaast leerde u dat de inhoud van een variabele niet alleen een getal hoeft te zijn, maar ook een reeks van lettertekens kan zijn. Zo'n reeks lettertekens wordt een string genoemd.
Om de computer duidelijk te maken wanneer een bepaalde letter, een bepaalde variabele, bedoeld is als naam voor een getal en wanneer een variabele bedoeld is als naam voor een string, plaatsen we ter onderscheiding een dollarteken achter de naam van een string.

Rekenen met strings en getallen

De computer beschouwt dat wat u hem aangeeft als string, niet als variabele of als te verwerken getal. De maximale lengte van een string is bepaald door de lengte van het geheugen. De minimumlengte van een string is nul tekens. Dit heet een nulstring of een lege string. U geeft dat als volgt aan:

```
LET A$ = ""
```

De aanhalingstekens worden niet meegeteld als horende bij de string. Ze geven slechts het begin en einde aan. Elke string die u gebruikt moet eerst gedimensioneerd worden. Zolang u dat niet doet, geeft het gebruik van een nog niet gedimensioneerde string *foutmelding nummer 9*. De string heeft door het dimensioneren een lege inhoud gekregen. Tik bijvoorbeeld:

```
DIM Q$(10)
PRINT Q$
```

De computer drukt niets op het scherm af en meldt daarna 'READY'. De inhoud van Q$ is leeg.
Omdat het dimensioneren van een stringvariabele moet gebeuren voordat u die stringvariabele verder in het programma gebruikt, is het een goede gewoonte alle, in het programma gebruikte, stringvariabelen in het begin van het programma te dimensioneren. Zie bijvoorbeeld regel 20 uit het vorige programma:

```
20 DIM N$(20),A$(25),P$(20),V$(10),T$(12)
```

Door deze regel kan in de rest van het programma de stringvariabele N$ maximaal 20 lettertekens bevatten. A$ kan 25 lettertekens bevatten, P$ kan maximaal 20 lettertekens krijgen, etc. Merk op hoe u in één regel slechts éénmaal de opdracht DIM hoeft te gebruiken. Alle te dimensioneren variabelen plaatst u, gescheiden door komma's, erachter.
Als u meer dan het gedimensioneerde aantal lettertekens aan een variabele opgeeft wordt het teveel afgekapt. Tik bijvoorbeeld:

```
DIM Z$(4)
Z$="teststring"
PRINT Z$
```

Hoewel u de computer opgegeven heeft dat Z$ gelijk is aan 'teststring', is de inhoud van Z$ volgens de computer gelijk aan 'test'. Dat zijn immers de eerste vier lettertekens van de string, en Z$ was gedimensioneerd tot 4 lettertekens.

Omdat elk letterteken in een string voor kan komen, is het goed er even bij stil te staan dat ook een getal een string kan zijn. Neem bijvoorbeeld het letterteken 7. U kunt dit teken op twee manieren bekijken:

als een getal: 7
als een string: "7"

U kunt van de ene manier van kijken naar de andere overschakelen. We nemen een string met de naam A$ en de inhoud van die string is het letterteken 7.

```
10 DIM A$(10):LET A$="7"
```

De twee manieren waarop we de 7 kunnen beschouwen, kunnen niet zomaar samen gebruikt worden. Het zal niet lukken de Atari de opdracht te geven:

```
20 PRINT A$+8
```

Reeds na het intikken van de regel volgt een foutmelding waarbij het plusteken als boosdoener wordt aangemerkt.

U heeft de computer eerst verteld dat de 7 een string was (met de naam A) en de computer heeft geleerd om strings niet te beschouwen als variabelen of getallen. *U heeft het type string en getal door elkaar gebruikt.* Dat past niet.
Het is echter wel mogelijk om strings in getallen om te zetten en andersom. Allereerst wordt elk karakter in de computer opgeslagen als een getal. Deze getallen stammen van de ASCII-set. Hierbij heeft elk lettertekens een bepaalde waarde. U vindt de waarden van de lettertekens in *bijlage 1* van dit boek. U kunt er ook een kort programma voor maken. De Atari kent een opdracht om de ASCII waarde van een letterteken als resultaat te geven. Dit is

ASC(..)

Tussen de haakjes plaatst u een letterteken of de naam van een string (een stringvariabele). Een programma dat de gebruiker telkens een letterteken laat intikken om vervolgens de ASCII waarde te geven is:

```
10 DIM A$(5):GRAPHICS 0
20 PRINT "KARAKTER: ";
30 INPUT A$
40 PRINT :PRINT "ASCII waarde van ";A$;" is ";ASC(A$)
50 PRINT :GOTO 20
```

U ziet dat in regel 40 bij de ASC opdracht geen aanhalingstekens staan. Dat is omdat de computer niet de ASCII waarde van de string 'A$' moet geven, maar de ASCII waarde van de string met de naam A$. Let u er ook op dat de ASC opdracht alleen de ASCII waarde van het eerste letterteken van een string geeft. Als u verschillende tekens achter elkaar intikt bij de INPUT krijgt u toch alleen de ASCII waarde van het eerste teken.
De andere kant op kan natuurlijk ook. U geeft de ASCII waarde op en de computer vertelt welk teken daarbij hoort.

CHR$(..) (CHaRacter=letterteken)

Met het volgende programma kunt u snel bijna alle tekens van de Atari zien.

```
10 GRAPHICS 0
20 FOR X=0 TO 255
25 IF X=125 THEN GOTO 40
30 PRINT CHR$(X);
40 NEXT X
```

Met een kleine aanpassing werkt dit programma ook in de andere tekstschermen.
Voor tekstscherm 1 verandert u regel 10 en 30:

```
10 GRAPHICS 1
30 PRINT #6;CHR$(X);
```

Ziet u dat u alleen letters krijgt, maar wel in twee verschillende kleuren? Tekstscherm 1 is alleen tekstscherm. De letters zijn groter, maar u heeft niet de beschikking over grafische tekens.
Voor tekstscherm 2 verandert u in dit laatste programma de 1 van de GRAPHICS opdracht in een 2 (om uit het programma te komen gebruikt u telkens de RESET knop). Dit programma eindigt in een foutmelding omdat er teveel lettertekens op het scherm geplaatst worden. Ook op dit scherm staan alleen lettertekens.
Op dezelfde wijze kunt u de tekstschermen 12 en 13 aanroepen. Probeer voor uzelf ook eens (als nieuwsgierig voorproefje) de grafische schermen (3 t/m 11, 14 en 15).
Het alphabet kent 26 letters. De Atari heeft dus 52 waarden nodig voor alle letters (hoofdletters en kleine letters). Daarnaast is nog een stel waarden nodig

voor allerlei tekens en de cijfers. De eerste 32 waarden zijn in de officiele ASCII-set in gebruik voor het aansturen van beeldscherm en randapparatuur. De Atari heeft deze codes tevens in gebruik voor allerlei grafische tekens. Om de Atari ASCII-set te kunnen onderscheiden van de officiele ASCII-set, wordt de set met de grafische tekens van de Atari ook wel **ATASCII genoemd**.

Voor ons probleem, het optellen van A$ (waarin zich '7' bevindt) en 8 geeft dit alles echter geen oplossing. Er is nog een stel opdrachten die het mogelijk maken om strings waarin zich alleen getallen bevinden om te zetten in de waarde van die getallen en andersom, om getallen om te zetten in strings met als inhoud die getallen. Om de waarde van een string te krijgen gebruikt u:

VAL(..) (VALue = waarde)

Als we dit toepassen op de string A$ ("7") tikken we:

LET X=VAL(A$)

De inhoud van A$ is '7', dus geeft VAL(A$) de waarde 7. De variabele X heeft als waarde 7 gekregen.
De VAL opdracht werkt alleen als tussen de haakjes een string (of de naam van een string) staat waarin getallen voorkomen. Het eerste lettertekens moet een plusteken (+) een minteken (-) of een cijfer zijn. Is dit niet zo, dan geeft de Atari een *foutmelding (nummer 18)*. Enkele voorbeelden:

```
DIM A$(10) : LET A$="88796"
LET X=VAL(A$)
PRINT X                     resultaat: 88796

LET X=VAL("-5456")
PRINT X                     resultaat: -5456

PRINT VAL("getal")          resultaat: ERROR- 18

PRINT VAL("3 getallen")     resultaat: 3

PRINT VAL("3*3=9")          resultaat: 3
```

Andersom kan natuurlijk ook. Daarvoor gebruikt u:

STR$(..)

Elke waarde die hier tussen haakjes geplaatst is, wordt tot string gemaakt. Natuurlijk kan ook een variabelenaam tussen de haakjes geplaatst worden. De waarde van de variabele wordt dan tot string gemaakt. Bijvoorbeeld:

```
LET X=654
LET A$=STR$(X)
PRINT A$                      resultaat: 654
```

Merk op dat in dit geval A$ niet gedimensioneerd wordt. Dat heeft u immers hiervoor al gedaan. De computer accepteert het niet als u een string tweemaal probeert te dimensioneren. U krijgt dan *foutmelding 9* te zien. U kun de dimensionering van een stringvariabele verwijderen door de BASIC opdracht

CLR

Met deze opdracht moet u echter voorzichtig zijn. Alle stringvariabelen worden gewist en moeten, als u ze weer wilt gebruiken, opnieuw gedimensioneerd worden. Daarnaast wist deze opdracht alle numerieke variabelen alsmede de array-variabelen (zie het hoofdstuk over arrays) en wordt de DATA wijzer hersteld (ook hier wordt nog op teruggekomen).

De optelling die we wilden maken van A$ (inhoud '7') met 8 lossen we als volgt op:

```
DIM A$(10)
LET A$="7"
PRINT VAL(A$)+8               resultaat: 15
```

Vergelijken van strings

Strings kunnen net als getallen met elkaar vergeleken worden. Daar worden dezelfde vergelijkingstekens voor gebruikt als in de rekenkunde.

=	gelijk aan
< >	niet gelijk aan
<	kleiner dan
>	groter dan
< =	kleiner dan of gelijk aan
> =	groter dan of gelijk aan

Deze vergelijkingen krijgen bij het gebruik van strings wel een eigen betekenis. **Twee strings zijn alleen gelijk (=) als ze even lang zijn en precies dezelfde lettertekens bevatten.** In alle andere gevallen zijn twee strings niet precies gelijk (< >). Strings worden letterteken vóór letterteken vergeleken. Een letterteken is groter dan het letterteken waarmee vergeleken wordt, als de ASCII waarde van de eerste hoger is. Een string is dus groter dan een andere string als het eerste letterteken groter is, dat wil zeggen als deze later in het alphabet of de ASCII codelijst staat (B > A, Z > A, F > B) en de string minstens zo lang is als de string waarmee vergeleken wordt. Zijn de eerste lettertekens gelijk, dan wordt gekeken naar het tweede teken. Zijn die ook gelijk, dan wordt gekeken naar het derde teken, enzovoort. Dit geschiedt op dezelfde wijze als in een woordenboek de woorden gerangschikt staan. Zijn alle lettertekens gelijk

dan wordt gekeken of een bepaalde string langer is. De langste string is de grootste (ABCD > ABC, XYZ > XY).
Een extra moeilijkheid is het verschil tussen hoofd- en kleine letters. Hoewel te verwachten is dat kleine letters kleiner zijn dan hoofdletters is dit niet zo. A < a, Ab < ab. Hoofdletters zijn kleiner dan kleine letters. U kunt dit zien aan de ASCII waarden. De ASCII waarden van de kleine letters zijn groter dan die van de hoofdletters. Zo zijn dus ook alle getallen kleiner dan de letters. Bij het alphabetiseren komen eerst de getallen, dan de hoofdletters en dan pas de kleine letters.
De overige vergelijkingstekens spreken verder voor zichzelf.
Heeft u een stel strings die afwisselend hoofdletters en kleine letters bevatten en wilt u deze toch op normale wijze alphabetiseren (zie ook het hoofdstuk 'Sorteren'), dan moet u op de een of andere wijze elke hoofdletter in een kleine letter omzetten of andersom. Daarvoor is het nodig dat u strings kunt manipuleren.

Manipuleren van strings

Verschillende computers kennen een hele reeks van opdrachten om strings te manipuleren. De Atari kent er slechts een, maar staat u met die ene opdracht dan ook ongekend veel toe. U kunt er elke string, ongeacht de lengte, mee verwerken.

```
STRINGVARIABELE(begin,eind)
```

Door twee getallen tussen haakjes achter de stringvariabele te zetten, kunt u elk deel van de string apart aanwijzen.
Bijvoorbeeld:

```
10 DIM A$(10),B$(10)
20 A$="boekje"
30 PRINT A$(1,3)            resultaat: boe
```

Het eerste getal tussen de haakje geeft de eerste te nemen letterpositie aan, het tweede getal de letterpositie van waar het laatste letterteken genomen moet worden. Verander regel 30 en RUN opnieuw:

```
30 PRINT A$(3,4)            resultaat: ek
```

Zoals u ziet kunt u ook uit het midden letters halen. Verander nogmaals:

```
30 PRINT A$(4,6)            resultaat: kje
```

Ook het meest rechtse deel van de string is opvraagbaar. Als u echter een letterpositie opgeeft die niet bestaat krijgt u ***foutmelding 5:***

```
30 PRINT A$(4,9)            resultaat: ERROR- 5
```

Behalve uitlezen van gedeelten van strings kunnen we op dezelfde wijze ook gedeelten van strings vullen:

```
10 DIM A$(10),B$(10)
20 A$="boekje"
30 A$(1,4)="boot"
40 PRINT A$                    resultaat: bootje
```

Let op wat er mis gaat als u vier letters wilt veranderen maar er slechts 3 opgeeft:

```
30 A$(1,4)="bot"               resultaat: botkje
```

De computer vult net zoveel letters in als hij opgekregen heeft, maar laat de andere onveranderd.
Probeer ook:

```
30 A$(1,4)=""                  resultaat: boekje
```

en

```
30 A$(1,4)="parkeren"          resultaat: parkje
```

Geeft u teveel letters op, dan kiest de computer vanaf het begin voldoende letters om het te veranderen gedeelte te vullen, en laat de rest zitten. Probeer uit wat er gebeurt als u een nieuwe regel 25 maakt:

```
25 A$="boe"                    resultaat: park
```

Hoe zou dit komen? A$ is in regel 25 toch maar 3 lettertekens lang gemaakt. Als het goed is, zou de opdracht in regel 30, namelijk om de eerste vier lettertekens te veranderen, tot een fout moeten leiden, omdat er slechts drie lettertekens aanwezig zijn.
Dat u toch de eerste vier lettertekens kunt vullen komt omdat u de maximale lengte van de string nog niet hebt gevuld. Deze is immers volgens regel 10, 10 karakters lang. Zo zou u ook kunnen invullen:

```
30 A$(1,10)="parkeen"          resultaat: parkeren
```

Als centrale regel dient u te onthouden dat bij het werken met strings de huidige lengte van een string altijd bepaald wordt door de laatst uitgevoerde stringopdracht voor die string.

Een andere zeer handige opdracht bij het werken met strings is een opdracht om de huidige (werkelijke) lengte van een string te bepalen:

LEN (..)

Tussen de haakjes plaatst u een string of de naam van een stringvariabele. De computer geeft vervolgens de lengte van de string weer:

```
DIM A$(10)
A$="boekje"
PRINT LEN(A$)                resultaat: 6
```

Palindromen

Een leuk programma dat strings manipuleert en vergelijkt is de zogenaamde palindroomtester. Een palindroom is een woord dat van voor naar achter en van achter naar voor gelezen hetzelfde is. Bijvoorbeeld het woord 'pap'. Maar er bestaan ook mooiere zoals: renner, avondnova of de beroemde parterreserretrap.
Dit programma laat u een woord invoeren, draait het om en controleert of het resultaat hetzelfde is.

```
10 GRAPHICS 1
20 DIM A$(50),Z$(50)
30 PRINT #6:PRINT #6;"WOORD a.u.b."
40 INPUT A$
50 PRINT #6:PRINT #6;A$
60 FOR X=LEN(A$) TO 1 STEP -1
70 Q=LEN(Z$)+1
80 Z$(Q,Q)=A$(X,X)
90 NEXT X
100 PRINT #6:PRINT #6;"omgekeerd: "
110 PRINT #6:PRINT #6;Z$
120 PRINT #6:PRINT #6:PRINT #6
130 IF Z$=A$ THEN PRINT #6;"EeN pAlInDrOoM !!"
140 FOR WA=1 TO 1500:NEXT WA
150 CLR :GOTO 10
```

De disk als notitieboek

Het volgende is bedoeld voor degene die een diskdrive heeft. Het is mogelijk de Atari als een eenvoudige tekstverwerker te gebruiken als u (al dan niet tijdelijk) geen ATARI-WRITER ter beschikking heeft. De truuk hiervoor bestaat uit het kopieren van het beeldscherm naar een diskette. Het volgende geldt voor DOS 2.0 en 2.5.
Kies van het DOS-menu eerst C van COPY (=kopieren), gevolgd door RETURN. De computer vraagt u dan van waar naar welke diskdrive u wilt kopieren en wat u wilt kopieren. U wilt van het toetsenbord naar de diskette kopieren. Laten we de tekst bijvoorbeeld WOORD noemen. Van het begin van het boek kunt u zich nog herinneren dat we BASIC programma's aangeven

met het achtervoegsel .BAS en ter onderscheiding worden teksten aangegeven met het achtervoegsel .TXT.
Het kopieren wordt dan:

E:,D:WOORD.TXT (De D: staat voor Disk, de E: staat voor Editor = opmaak)

Voordat u op RETURN druk moet u eerst de diskette waar u de tekst op wilt bewaren in de diskdrive plaatsen.
Na RETURN gaat uw diskdrive een tijdje draaien.
Zodra uw diskdrive stil staat kunt u beginnen met tikken. U kunt elke zin verbeteren zolang u de zin nog niet afgesloten heeft met RETURN. Daarna is verbeteren niet meer mogelijk. U kunt het dus het beste vergelijken met een elektrische schrijfmachine met een correctietoets voor een regel.
Zodra u klaar bent met schrijven drukt u *tegelijk* op CONTROL en de 3-toets. Uw tekst wordt nu op de diskette weggezet.
Om uw tekst nog eens te zien, gebruikt u weer de COPY funktie, maar nu kopieert u van diskette naar scherm: kies weer C van COPY en tik vervolgens:

D:WOORD.TXT,S: (De S: staat voor Screen = scherm)

Als uw tekst te snel voorbij flitst kunt u gebruik maken van de CONTROL en de 1-toets om het rollen van het beeld stil te zetten. Drukt u nog één keer deze twee toetsen in, dan spoedt de tekst zich weer verder over het scherm.
Om de tekst naar een printer te sturen kiest u ook de COPY-funktie van het DOS-menu en tikt u:

D:WOORD.TXT,P: (de P: staat voor Printer)

De gebruikers van DOS 3.0 moeten iets anders te werk gaan. Het principe blijft hetzelfde.
Kies van het menu de C voor COPY-funktie.
Geeft op de vraag 'APPEND (Y/N)' het antwoord N (van No).
De computer vraagt om 'source device?' (= bron apparaat). U antwoordt met E: (van het toetsenbord).
De computer vraagt om 'destination device?" (=bestemmings apparaat). U antwoordt met D: (van de diskdrive).
Als de computer om een 'destination filename?' vraagt vult u, net als hierboven bijvoorbeeld, WOORD.TXT in.
De computer geeft als laatste aanwijzing: 'Insert destination disk and press RETURN (=stop de bestemmingsdiskette in de diskdrive en druk op RETURN). Nu kunt u de diskette waar u uw tekst op wilt bewaren in de diskdrive plaatsen.
Verder gaat het als hierboven beschreven. Het enige verschil tussen de beide DOS systemen is dat DOS 3.0 de vragen wat meer uit elkaar rekt, zodat u vaker een kort antwoord moet geven in plaats van één enkele keer een lang antwoord.

10. LIJSTEN MET GEGEVENS

Inleiding

De tot nu toe in dit boek behandelde programma's werkten meestal met slechts enkele gegevens. Soms moest de computer zelf gegevens maken of een bepaald gegeven varieren.
Naast hun rekenfunktie staan computers echter vooral bekend om hun mogelijkheden tot het verwerken van grote hoeveelheden gegevens in tamelijk korte tijd. Dit hoofdstuk laat zien op welke manier deze gegevens aan de computer opgegeven kunnen worden en hoe de computer deze gegevens kan opslaan.
We gebruikten hiervoor de INPUT of LET opdracht . De INPUT opdracht zorgt ervoor dat de computer tijdens het programma om een gegeven vraagt. Dit gegeven wordt toegekend aan een variabele en daarmee gaat de computer verder aan het werk. De LET opdracht zorgt ervoor dat een waarde aan een variabele wordt toegekend. Beide methoden hebben het nadeel dat het slechts om beperkte hoeveelheden gegevens gaat. Daarbij komt nog het bezwaar dat de gegevens, zoals die tot nu toe via de INPUT opdracht werden ingevoerd, slechts een programma-run lang blijven bestaan.
Een eenvoudige manier om een stel gegevens aan de computer op te geven is door middel van een DATA (gegevens) lijst.

DATA-lijsten

Met de BASIC opdracht

DATA

kunt u een lijst gegevens aan de computer opgeven. De gegevens worden onderling gescheiden door **komma's.** Zowel getallen als strings zijn mogelijk. In de DATA opdracht hoeven de strings niet geopend en gesloten te worden door aanhalingstekens. Als er echter komma's, dubbele punten of spaties in de string voorkomen moeten deze wel door aanhalingstekens omgeven worden. Strings met en strings zonder aanhalingstekens kunnen door elkaar in een DATA opdracht worden gebruikt.
Het is wel nodig dat de bij een string behorende leesopdracht (zie hieronder) aan de computer duidelijk maakt dat het om een string gaat. Een lijst kan meer dan één programmaregel beslaan. In dat geval moet aan het begin van elke regel de DATA opdracht herhaald worden. Ondanks een verdeling over verschillende regels ziet de computer de lijst toch als één geheel.
De computer kan de gegevens van de lijst lezen door middel van de opdracht:

READ

De gegevens worden gelezen in de volgorde waarin ze in de datalijst staan, van links naar rechts, van laag naar hoog regelnummer. Achter de READ wordt de naam gegeven van de variabele waaraan het gegeven toegekend dient te worden. Bijvoorbeeld:

```
10 READ A
20 PRINT A
30 DATA 318, 645, 75
```

of

```
10 DIM A$(20)
20 READ A$
30 PRINT A$
40 DATA voorbeeldsliert, tekst, tekst
```

Uit dit voorbeeld blijkt al dat de DATA en de READ opdracht niet bij elkaar hoeven te staan. **Het is de gewoonte de datalijst aan het einde van het programma te plaatsen.** Dit verhoogt de leesbaarheid van het programma. Deze werkwijze levert, op bovenstaande manier gebruikt, niet zoveel op, maar het is mogelijk aan verschillende variabelen waarden toe te kennen door middel van de datalijst:

```
10 READ A,B,C
20 PRINT A,A*B,C
30 DATA 10,6,80,90,120
```

De computer houdt zelf bij waar hij in de datalijst gebleven is. Daardoor kunt u met één datalijst volstaan, terwijl op verschillende plaatsen in het programma naar deze lijst verwezen wordt:

```
10 READ A,B,C
20 PRINT A,A*B,C
:
:
70 READ D,E
80 PRINT D*E,D,E*E
:
:
200 DATA 1,2,3,4,5,6,7,8,9
```

In dit geval zijn alle gegevens in één programmaregel gezet. Programma's waarin verschillende keren verwezen wordt naar een lange datalijst zijn beter leesbaar als u elke bij elkaar horende groep gegevens in een aparte dataregel zet. Dit maakt terugzoeken eenvoudiger.
Dat de computer zo aardig is om de tel bij te houden in de datalijst, maakt het programmeren voor ons gemakkelijker. Stel dat we een eenvoudige lijst met telefoonnummers willen maken. Als we een naam intikken, moet de computer het bijhorende telefoonnummer voor ons opzoeken. In onderstaand programma zijn alle gegevens strings.

```
10 GRAPHICS 0
20 DIM NA$(10),DA$(10),TE$(16)
30 PRINT "NAAM: ";:INPUT NA$
40 PRINT
50 READ DA$,TE$
60 IF DA$="EINDELIJST" THEN PRINT "KOMT NIET IN LIJST VOOR":END
70 IF NA$<>DA$ THEN GOTO 40
80 PRINT TE$
90 END
200 DATA JAN,434423,PIET,0453-8876,GIJS,765453,MARIE,078-987876
205 DATA KLAARTJE,243251
210 DATA KEES,020-565354,TRUUS,766545,SJEF,0931-766754/7654
215 DATA GEERT,03454-9786,EINDELIJST,XXX
```

In dit programma leest de computer elke keer een naam uit de lijst (DA$) en een bijhorend telefoonnummer (TE$). Hij doet dit net zo lang tot de opgegeven string gelijk is aan een in de lijst voorkomende naam (IF NA$ < > DA$ THEN GOTO 40). Dan springt de computer uit de lus en geeft het telefoonnummer weer. Komt de computer bij de laatste naam van de lijst (einde lijst) dan is het duidelijk dat de naam niet in de lijst voorkomt. Dit moet gemeld worden en de computer moet het programma beeindigen.
Het voordeel van zo'n programma is dat u een eindeloze lijst namen met bijhorende telefoonnummers kunt intikken en dan de computer een telefoonnummer bij een bepaalde naam kunt laten opzoeken. Let u bij bovenstaand programma op dat er verschil is tussen een hoofdletter en een kleine letter.
Voor de Atari is Jan een ander dan JAN of dan jan.
Het is ook mogelijk een lijst met gegevens verschillende keren te gebruiken. Daarvoor kent de Atari de opdracht:

RESTORE (herstel)

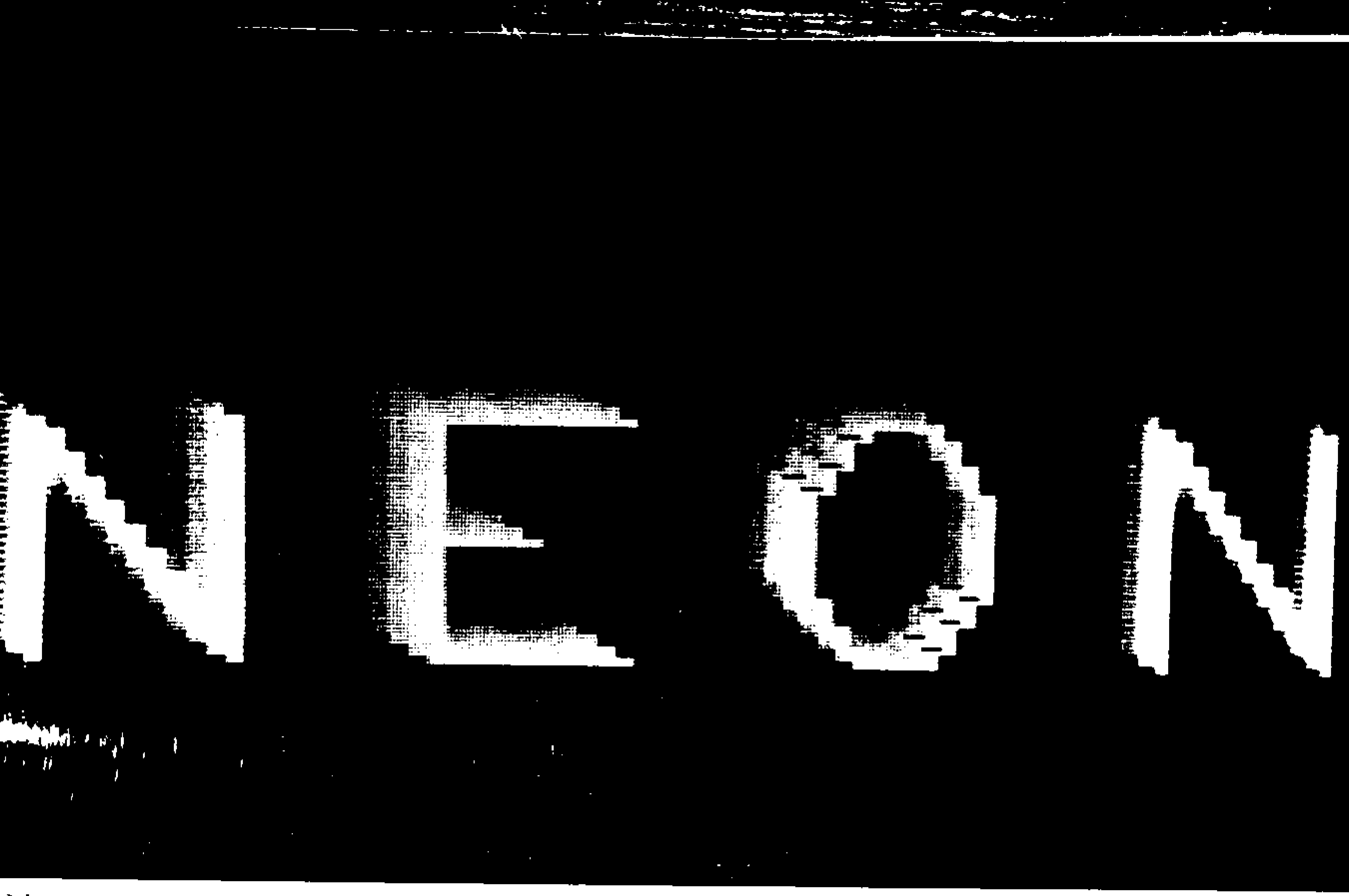
Deze opdracht beinvloedt de plaats in het geheugen van de Atari, waar een pijltje bij de gegevens uit de lijst staat. Dit pijltje, een zogenaamde datawijzer (gegevenwijzer), zorgt ervoor dat de computer weet waar hij in de lijst is gebleven. RESTORE zonder enige toevoeging zorgt ervoor dat de datawijzer terugspringt naar het eerste gegeven achter de eerste DATA- opdracht in een programma.

```
10 READ A,B,C
20 PRINT A,B,C
:
:
40 RESTORE
50 READ D,E,F
60 PRINT D,E,F
:
:
100 DATA 1,2,3,4,5,6
```

De waarden van A, B en C zullen respectievelijk 1, 2 en 3 zijn.

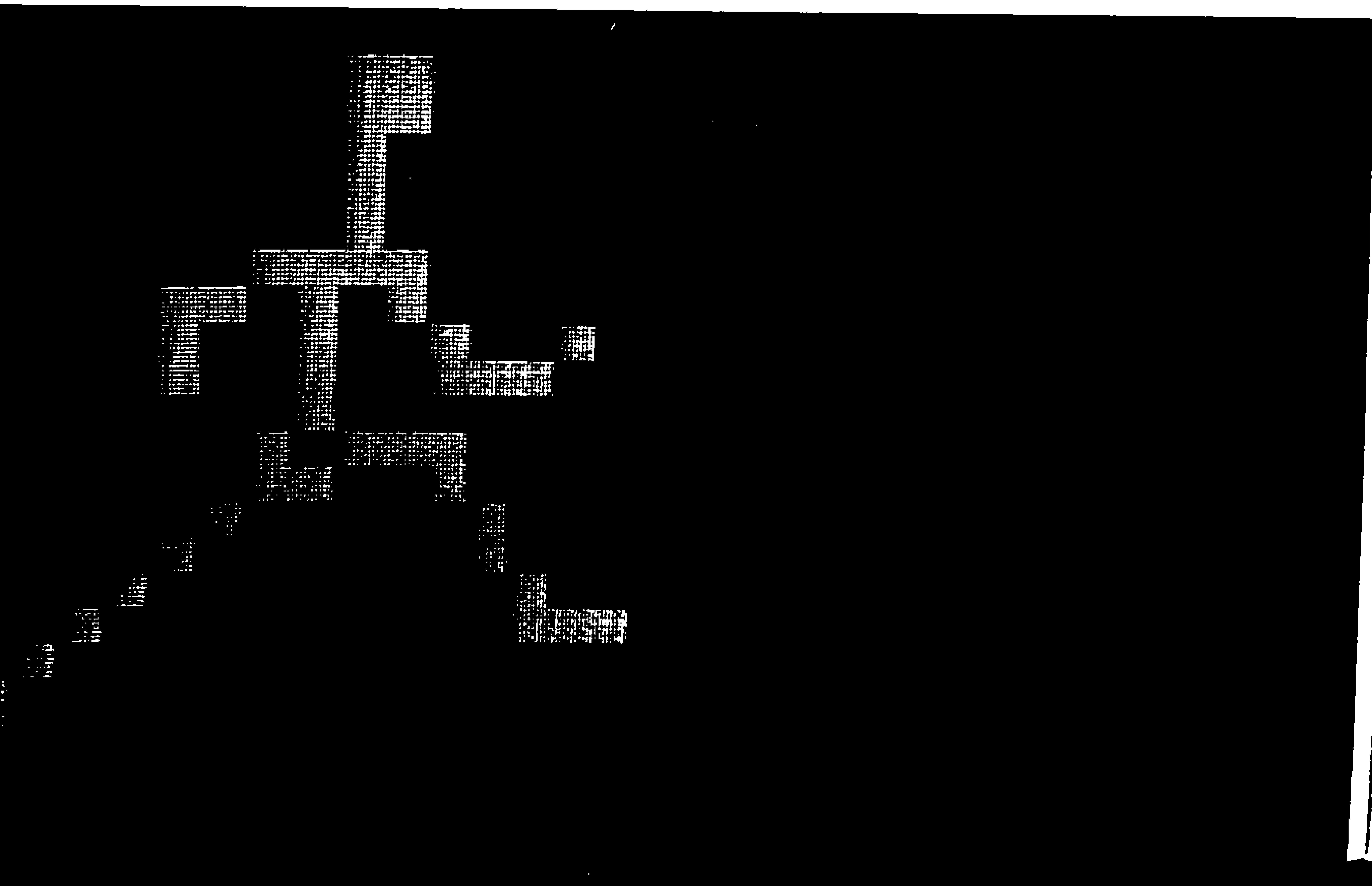
De waarden van D, E en F zijn ook 1, 2 en 3. De datawijzer is door de opdracht in regel 40 teruggesprongen naar het eerste gegeven.
Achter RESTORE kan ook een regelnummer opgegeven worden. In dat geval springt de datawijzer naar het eerste gegeven achter de DATA opdracht op de opgegeven regel.
RESTORE hoeft niet alleen gebruikt te worden om de computer terug te laten gaan naar gegevens die hij al eerder gebruikt heeft. Het is door middel van het opgeven van een regelnummer ook mogelijk om een stel gegevens (al dan niet tijdelijk) over te slaan.
RESTORE geeft eveneens de mogelijkheid om te kiezen tussen verschillende datalijsten. In het volgende programma zorgt elke datalijst voor woorden om uw talenkennis op te frissen. De gebruiker kan kiezen welke lijst hij wil gebruiken.

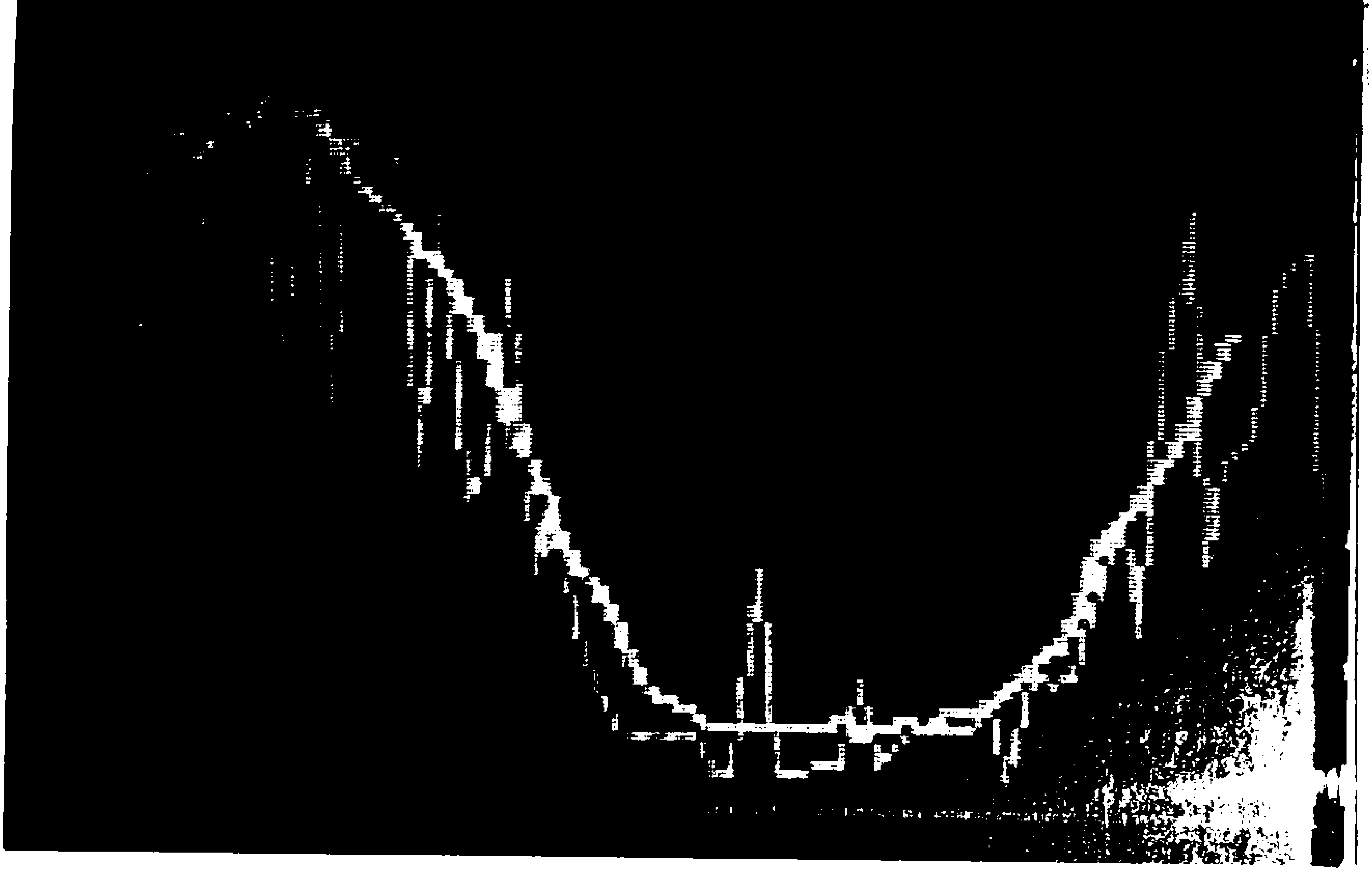
```
10 DIM A$(20),N$(20),V$(20)
20 GRAPHICS 0
30 PRINT "     ***********      "
40 PRINT "      TALENTEST     "
50 PRINT "       *******"
60 PRINT :PRINT "U kunt kiezen uit: ":PRINT
70 PRINT "1- NED.  >> DUITS"
80 PRINT "2- DUITS >> NED."
90 PRINT "3- NED.  >> FRANS"
100 PRINT "4- FRANS >> NED."
110 PRINT "5- NED.  >> ENGELS"
120 PRINT "6- ENGELS>> NED."
130 PRINT "7- NED.  >> ZWEEDS"
140 PRINT "8- ZWEEDS>> NED."
150 PRINT
160 PRINT "Uw keuze ";:INPUT K
170 IF K=1 OR K=2 THEN RESTORE 520
180 IF K=3 OR K=4 THEN RESTORE 530
190 IF K=5 OR K=6 THEN RESTORE 540
200 IF K=7 OR K=8 THEN RESTORE 550
210 GRAPHICS 0:T=0
220 FOR X=1 TO 4
230 READ N$,V$
240 IF INT(K/2)=K/2 THEN 300
250 PRINT :PRINT N$
260 PRINT "Uw vertaling:";:INPUT A$
270 IF A$=V$ THEN T=T+1:PRINT "PRIMA!":GOTO 280
275 PRINT "JAMMER, FOUT"
280 NEXT X
290 GOTO 340
300 PRINT :PRINT V$
310 PRINT "Uw vertaling:";:INPUT A$
320 IF A$=N$ THEN T=T+1:PRINT "PRIMA!":GOTO 330
325 PRINT "JAMMER, FOUT"
330 NEXT X
340 PRINT :PRINT T;" goede antwoorden"
350 END
520 DATA HEBBEN,HABEN,NIEUW,NEU,SCHADUW,SCHATTEN,AREND,ADLER
530 DATA BRAND,FEU,HUIS,MAISON,GEBIT,DENTURE,TAFEL,TABLE
540 DATA HOOG,HIGH,BOEK,BOOK,HUIS,HOUSE,BRIEF,LETTER
550 DATA VADER,FAR,GEVAAR,FARA,AUTO,BIL,DIEP,DJUP
```



Neon programma *blz. 161*

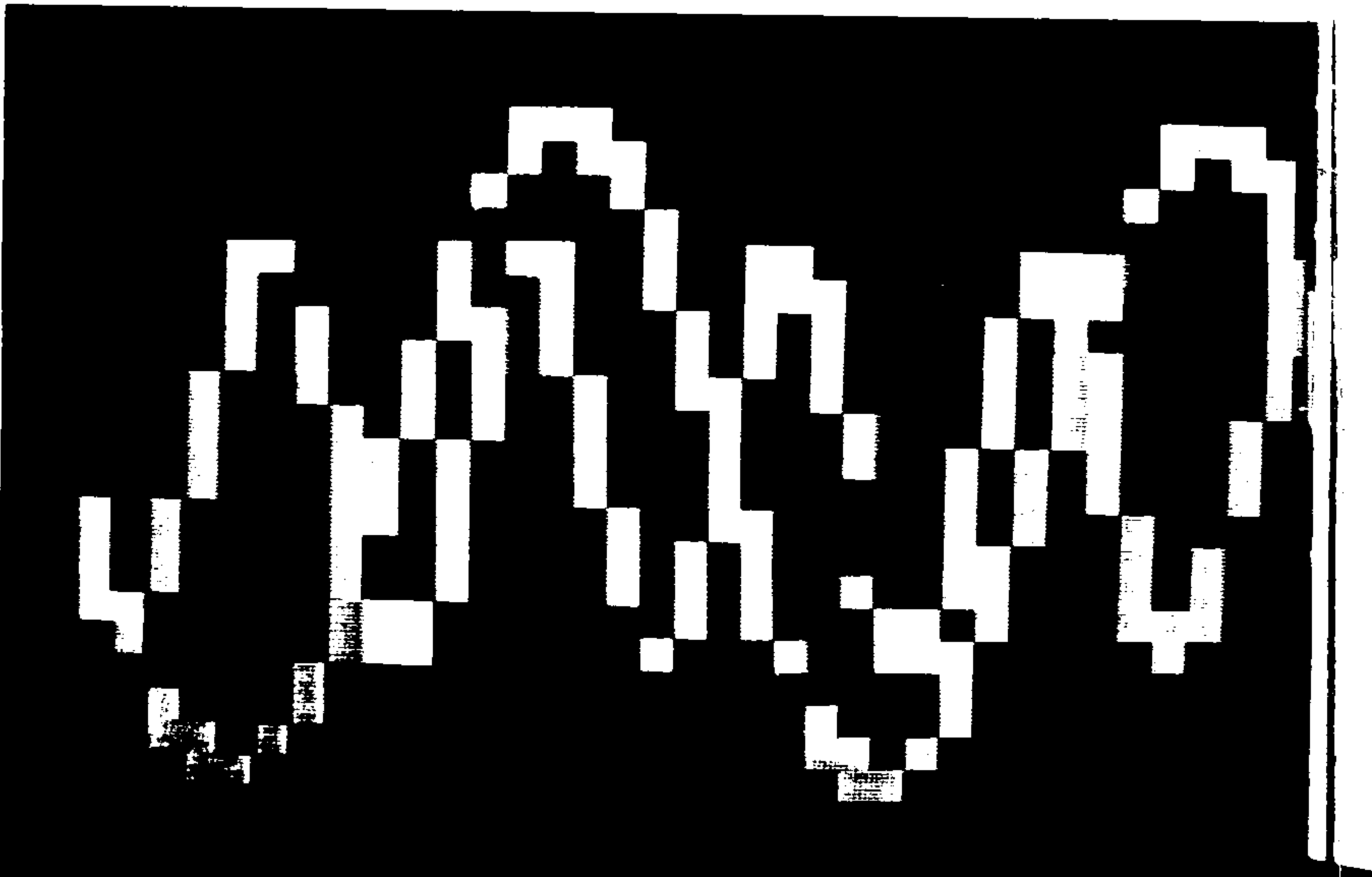
Mannetje programma [illegible]





De energierekening programma *blz. 102*

Twee sinussen en een cosinusfunktie grafiek programma *blz. 190*



Dit is een computerboek en geen talenboek , dus is de gegevenslijst zeer kort. U kunt hem zelf zo lang maken als u wilt. Als u net als in het voorbeeldprogramma meer dan één taal in de gegevenslijst wilt hebben staan, kunt u nog verder bekorten door telkens maar één keer een nederlands woord te plaatsen met daarachter de vertalingen in alle gewenste talen. Probeert u zelf te verzinnen hoe u dan het juiste woord uit de lijst haalt en hoe u daarna weer zorgt voor het lezen van een nederlands woord. Het is wat gepuzzel, maar u moet eruit kunnen komen.

Arrays

Met de READ en DATA opdracht is het mogelijk lange lijsten van gegevens of waarden aan de computer op te geven en deze toe te kennen aan een stel variabelen of telkens aan dezelfde variabele. Soms moet u een stel gegevens tegelijk verwerken en niet zoals in de voorbeeldprogramma's alleen even doorlopen om de juiste er uit te zoeken. Voor zo'n lange lijst met gegevens kunnen we gewoon een heleboel variabelen kiezen waaraan we de gegevens toekennen. Maar als het gaat om een boel gegevens, wordt het nogal lastig om al die variabelen uit elkaar te houden en te onthouden welke variabele wat voorstelt.
Daarom kent de Atari de mogelijkheid om lijsten met genummerde variabelen te maken. Zo'n lijst heet een ARRAY (=tabel). Variabelen die in een array geplaatst zijn, zijn gemakkelijk te hanteren en men kan er eenvoudig naar verwijzen. De enige voorwaarde voor het gebruik van een array is dat u de computer van te voren vertelt hoeveel geheugenruimte er voor de lijst gereserveerd moet worden . Dit doet u door middel van de opdracht:

DIM (DIMension = afmeting)

Deze opdracht kent u nog van het vorige hoofdstuk waar hij gebruikt werd om geheugenruimte te reserveren voor strings. De funktie is in beide gevallen hetzelfde, namelijk het voor een bepaald doel vastzetten van een gedeelte van het geheugen. Alleen dient dit aparte geheugendeel bij de stringtoepassing voor het opslaan van strings, en bij deze toepassing voor het opslaan van willekeurige numerieke gegevens in een array.

Een-dimensionale arrays.

Stel u wilt een lijst maken van 40 gegevens. U krijgt dan te maken met de volgende problemen:

- Hoe kan deze lijst onderscheiden worden van een andere soortgelijke lijst?
- Hoe kunnen gegevens van de lijst onderling onderscheiden worden?
- Hoe moet naar een bepaald gegeven in de lijst verwezen worden?
- Hoeveel geheugenruimte moet er gereserveerd worden?
- Hoe worden de gegevens in de lijst geplaatst?

De eerste vraag is het eenvoudigst te beantwoorden. Elke lijst krijgt een eigen naam. Als eerste naam kiezen we bijvoorbeeld L (van lijst). De eenvoudigste manier om een bepaald gegeven in de lijst aan te geven is de gegevens te nummeren. Er kan dan verwezen worden naar het eerste, het tweede, het derde, het twaalfde of het X-ste gegeven van de lijst L.
De notatie hiervoor is:

L(1) - eerste gegeven in de lijst L
L(2) - tweede gegeven in de lijst L
L(X) - het X-de gegeven in de lijst L (als er niet eerder in de programma-loop een waarde aan X is toegekend, is X gelijk aan 0)

Op deze wijze is elk gegeven in elke lijst aan te geven.
De DIM opdracht wordt gebruikt om voldoende geheugenruimte te reserveren. U hoeft niet zelf uit te rekenen hoeveel dit moet zijn. U geeft achter DIM alleen tussen haakjes de lengte van de lijst weer. Bijvoorbeeld:

DIM L(40)

voor een lijst met 40 gegevens.
Om de lijst met gegevens te vullen kan de LET-opdracht gebruikt worden:

```
10 DIM L(40)
20 LET L(1)=2
30 LET L(2)=4
40 LET L(3)=5
:
:
410 LET L(40)=650
```

Maar dit kost veel intiktijd. Het is juist zo eenvoudig te verwijzen naar een bepaald gegeven in de lijst. U kunt zich uit het vorige stuk vast nog wel de datalijst herinneren. Veel beter is dan ook:

```
10 DIM L(40)
20 FOR X=1 TO 40
30 READ D : L(X)=D
40 NEXT X
50 DATA 2,4,5..............................650
```

Dit bespaart enkele honderden programmaregels.
Let op de wat vreemde constructie in regel 30. Het is niet mogelijk direkt een waarde in een array variabele te lezen door middel van READ. De waarde wordt daarom eerst in een gewone variabele (D) geplaatst en vervolgens wordt L(X) gelijk gemaakt aan die variabele. Dezelfde constructie is nodig als u een INPUT opdracht wilt gebruiken voor het invullen van een array-variabele.

Twee-dimensionale arrays

Soms moet er niet een lange lijst met getallen verwerkt worden, maar een hele bladzijde. De getallen hebben dan (soms) niet alleen in een bepaalde richting iets met elkaar te maken, maar ook in de richting haaks daarop. De getallen staan in, met elkaar te maken hebbende, kolommen en rijen. Voor de normale lijst van gegevens achter elkaar wordt een zogenaamde één-dimensionale array gebruikt. De eerste dimensie wordt voorgesteld als een lijn. Een dergelijke één-dimensionale array wordt ook wel een vector genoemd.
Voor een bladzijde met getallen wordt een twee-dimensionale array gebruikt. De tweede dimensie wordt voorgesteld als een plat vlak. Een andere naam voor zo'n array is een tabel. In een twee-dimensionale array staan de gegevens in rijen en kolommen gerangschikt. Een twee-dimensionale array met 25 gegevens ziet er bijvoorbeeld zo uit:

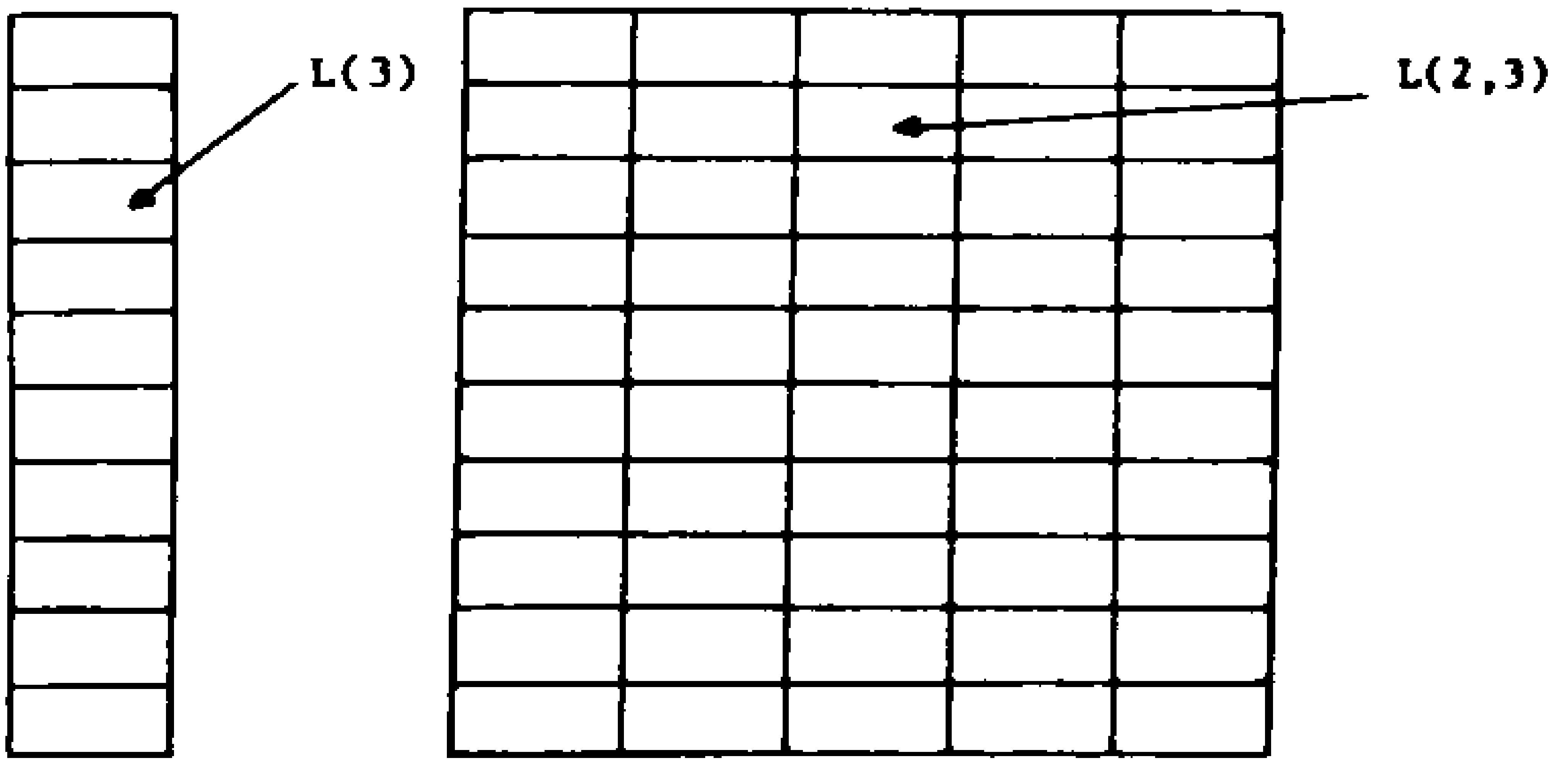
2	4	6	8	10
12	14	16	18	20
22	24	1	3	5
7	9	11	13	15
17	19	21	23	25

Het verwijzen naar een bepaald gegeven in deze array is iets lastiger. U noemt nu niet meer het zoveelste gegeven in de lijst, maar u verwijst naar de zoveelste regel, zoveelste kolom:

L(3,2) is het tweede gegeven van de derde regel (24)
L(2,4) is het vierde gegeven van de tweede regel (18)
L(X,Y) is het Y-de gegeven van de X-de regel, of regel X, kolom Y

Het reserveren van voldoende ruimte in het geheugen voor een twee-dimensionale array gaat met **twee** getallen:

DIM L(5,5)



een- en tweedimensionale array

Het is niet nodig dat het array vierkant is. Als u bijvoorbeeld 100 gegevens op wilt bergen in een array van 4 kolommen van elk 25 gegevens of 25 rijen van elk 4 gegevens maakt u het volgende programma:

```
10 DIM K(25,4)
20 FOR Y=1 TO 4
30 FOR X=1 TO 25
40 READ Z : K(X,Y)=Z
50 NEXT X : NEXT Y
60 DATA ga1, ga2, ga3, ga4, .., .., ......................., ga25
70 DATA gb1, gb2, gb3, gb4, .., .., ......................., gb25
80 DATA gc1, ................................................, gc25
90 DATA ....................................................., gd25
```

Let u hierbij op de volgende punten:

- U kunt zelf kiezen of u de gegevens per kolom of per regel laat lezen. In het voorbeeld hierboven wordt per kolom gelezen.
- Houdt u bij het intikken van de datalijst goed in de gaten in welke volgorde er ingelezen wordt.
- Het verhoogt de leesbaarheid en vermindert de kans op fouten als u de gegevens in de datalijst per dataregel logisch bij elkaar zet. In het voorbeeld werden alle gegevens van een kolom in een dataregel bij elkaar gezet. Natuurlijk kunnen de gegevens ook per rij ingevoerd worden.
- Een onderverdeling van de datalijst in verschillende regels is ook handig als bepaalde delen uit de lijst verschillende keren gebruikt moeten worden. De RESTORE opdracht kan alleen naar een bepaalde regel verwijzen.

Werken met arrays

De variabelen in een array worden niet anders behandeld dan andere variabelen. Dat wil zeggen dat u er alle logische of rekenkundige bewerkingen en vergelijkingen op los kunt laten. Alle arrays die we tot nu toe hebben laten zien waren zogenaamde numerieke arrays. Dat wil zeggen dat de array bestond uit een verzameling variabelen waarin getallen opgeslagen kunnen worden.
De Atari kent geen mogelijkheid om strings in een array op te slaan. Omdat het toch af en toe handig kan zijn om een array met strings te hebben, vindt u hier een korte beschrijving hoe u dit probleem kunt omzeilen. Heeft u de eerste tijd geen string-arrays nodig, sla dit stuk dan tijdelijk over. Lees het na zodra u er behoefte aan heeft.
Voor stringarrays geldt hetzelfde als voor numerieke arrays: ze zijn zo handig omdat door het slechts noemen van een getal naar een bepaalde variabele verwezen kan worden.
Omdat de Atari geen stringarrays kent, maar wel een krachtige methode heeft om delen van een string aan te wijzen (zie vorige hoofdstuk), is het mogelijk een zeer lange string te maken en deze als array te behandelen.
We maken als het ware een array waarvan alle elementen op een lange rij zijn gezet. Met een eenvoudige formule vertellen we de computer vervolgens waar

hij in de lange hoofdstring moet zoeken naar een bepaalde substring. Stel u wilt een een-dimensionale stringarray van 20 elementen waarbij elk element 30 karaktertekens heeft. Daarvoor maken we een hoofdstring van 600 lettertekens. Het eerste element is te vinden op de plaatsen 1 tot en met 30, het tweede op de plaatsen 31 tot en met 60, het derde op de plaatsen 61 tot en met 90, enzovoorts. Het element X bevindt zich op de plaatsen:

(30*X)+1 tot en met 30*(X+1)

Let hierbij op de telling: X loopt in het voorbeeld van 0 tot en met 19. Element 0 is het eerste element. Element 1 is het tweede element, element X is het X+1-de element. Het is net als tellen terwijl u met 0 begint te tellen.
De voordelen zijn duidelijk: niet alleen kan elk element nu snel opgezocht worden, maar met behulp van de in het vorige hoofdstuk geleerde stringmanipulaties is het mogelijk een bepaald element te veranderen. Tevens kunnen de verschillende elementen gewisseld worden, waardoor het mogelijk is te sorteren naar alphabet.
Het nadeel van deze methode is dat het tamelijk veel ruimte kost, omdat niet elke string altijd 30 karakters lang zal zijn. De ruimte is er echter voor gereserveerd, dus kost het geheugenruimte. Het is mogelijk om dit nadeel te verhelpen. Naast de stringarray maakt u dan tevens een numerieke array. In de numerieke array plaatst u de gegevens over de lengtes van alle deelstrings (door middel van LEN). Bij het manipuleren van de deelstrings maakt u telkens gebruik van de gegevens in de numerieke array om de juiste plaats voor de deelstrings op te zoeken. Aan deze methode zitten echter veel haken en ogen. Zo wordt het extra lastig als u een korte string wilt vervangen door een langere. Alle andere deelstrings erachter moeten dan opschuiven. Er is immers geen vrije reserveruimte tussen de strings. Zolang u geen echt lange programma's maakt die erg veel geheugenruimte vragen is het beter de hierboven beschreven methode te volgen.
In het volgende hoofdstuk vindt u een voorbeeld van het gebruik van een stringarray.

De energierekening

Het volgende programma laat u zien hoe een gegevenslijst omgezet kan worden in een mooie grafiek. Gedurende een jaar is elke week de stand van de gasmeter opgenomen. Daardoor viel het wekelijkse verbruik uit te rekenen. Door alle verbruikcijfers onder elkaar te zetten zou een lange lijst met getallen ontstaan waarvan de lezer niet veel wijzer zou worden. Door de verbruikcijfers te verwerken in een grafiekje dat het verbruik per week afzet tegen de tijd (de weken), is in een oogopslag te zien hoe het energieverbruik zich door het jaar heen ontwikkelt. De verbruikcijfers zijn in een stel dataregels opgeslagen.

```
10 GRAPHICS 15+16
20 COLOR 1
30 PLOT 0,180
40 FOR A=0 TO 50 STEP 3
50 DRAWTO A*3+5,180
60 PLOT A*3+5,190
70 DRAWTO A*3+5,180
80 NEXT A
90 REM
100 PLOT 5,180
110 FOR A=0 TO 180 STEP 10
120 DRAWTO 5,180-A
130 PLOT 2,180-A
140 DRAWTO 5,180-A
150 NEXT A
160 REM
170 RESTORE 400
180 COLOR 2
190 PLOT 5,72
200 FOR A=1 TO 47
210 READ C
220 DRAWTO A*3+5,180-C
230 NEXT A
240 REM
250 RESTORE 500
260 COLOR 3
270 PLOT 5,100
280 FOR A=1 TO 47
290 READ E
300 DRAWTO A*3+5,180-E
310 NEXT A
320 GOTO 320
400 DATA 137,107,77,93,88,76,102,136,149,119,88,100
410 DATA 77,120,120,120,64,53,58,63,50,44,55,34,18,14,14,14
420 DATA 13,7,7,6,7,7,9,9,8,14,15,16,16,20,15,28,23,25
430 REM
500 DATA 88,74,73,84,64,74,97,82,101,116,65,71,88,66,66,66
510 DATA 79,66,66,68,66,65,77,63,55,43,43,43,49,46,46,46,46,46,46
520 DATA 55,25,36,50,62,77,54,112,80,87,85,143
```

De regels 20-160 zorgen ervoor dat langs de verticale lijn links op het scherm een verdeling komt te staan voor de verbruikte hoeveelheid gas in een week (kubieke meters). Langs de horizontale lijn beneden op het scherm komen streepjes voor de weken te staan. Voor elke week een streepje.
De verschillende opdrachten in dit programma zijn nog niet behandeld. U kunt hiervoor de hoofdstukken over grafiek en kleur lezen. Beter is het om later als u dat deel gelezen heeft, dit programma nog eens te bekijken.
De regels 170-230 lezen de gegevens die bij het gasverbruik horen uit de datalijst. Deze gegevens staan in de regel 400-420.
De regels 250-310 doen precies hetzelfde voor de electragegevens. De plaats waar de lijn van de grafiek naartoe getrokken moet worden is in de hoogte afhankelijk van een verbruikscijfer uit de datalijst, en in de breedte afhankelijk van de grootte van A (A*3+5). Regel 320 zorgt ervoor dat de grafiek niet onmiddellijk na het tekenen weer verdwijnt. ***Probeert u deze regel weg te halen en let op het effekt.***

De regels 400 tot en met 520 bevatten de verbruikscijfers. Voor elke week een. In dit programma ligt van te voren vast dat het aantal gegevens precies voldoende zal zijn voor een jaar. Maar het kan dat u het programma regelmatig wilt veranderen, zodat u elke week een nieuwe bijgewerkte grafiek kunt zien. Daarvoor moet u A in regel 140 niet van 1 tot 51 laten lopen, maar van 1 tot Z. De waarde van Z is dan afhankelijk van het aantal data in de datalijst. Dat kunt u aangeven door als eerste gegeven in de datalijst het aantal gegevens in de datalijst op te geven. Bijvoorbeeld door een nieuwe regel 390 te maken:

```
390 DATA 51 : REM aantal gegevens in datalijst
```

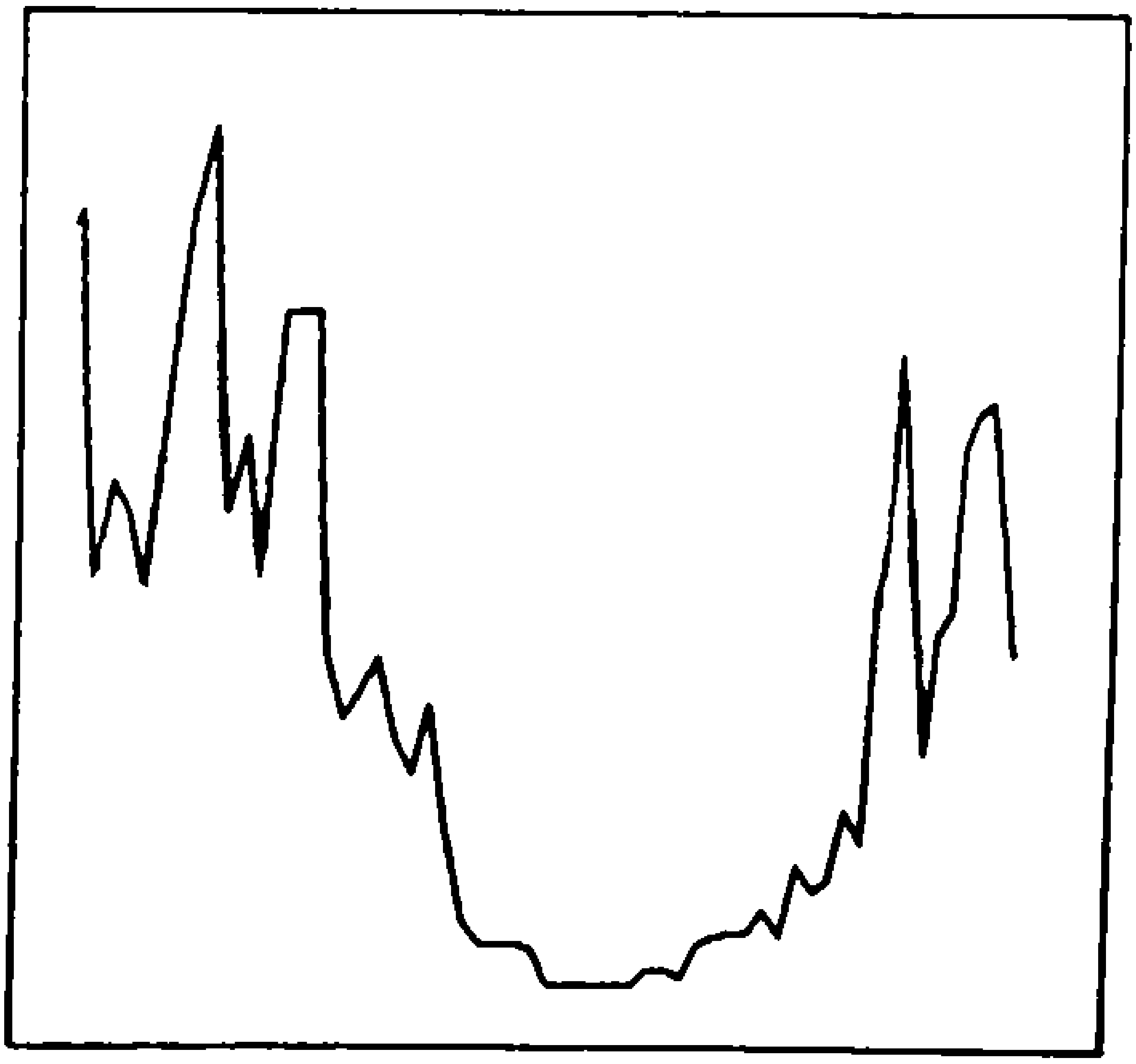
Aan Z moet nu de juiste waarde toegekend worden door een lees (READ) opdracht. Maak een regel 165:

```
165 READ Z : REM aantal gegevens in datalijst
```

Tot slot hoeft alleen regel 200 veranderd te worden:

```
200 FOR A=1 TO Z
```

U kunt in dit programma eenvoudig andere gegevens invullen. U past daarvoor de datalijst aan. Bekijk op deze wijze uw eigen energieverbruik door het jaar heen of zet hele andere gegevens in de grafiek. Als u het oorspronkelijke programma eerst SAVEt op een cassete of diskette kunt u naar hartelust experimenteren zonder angst voor het vernielen van wat tikwerk.



11. SORTEREN

Bij het sorteren van lijsten met gegevens is het onontbeerlijk om de gegevens tijdelijk op te kunnen slaan in een array. De gegevens worden daarna op volgorde in de array geplaatst. De computer maakt bij het sorteren gebruik van de nummering van variabelen in een array.
Een sorteerprogramma kan voor vele toepassingen handig zijn. U kunt er lange lijsten met getallen mee op volgorde zetten of u kunt er bestanden met adressen, cassettebandjes, diskettes, platen, computerprogramma's, enzovoort, mee op alphabetische volgorde zetten.

Wat gebeurt er bij het sorteren van een lijst getallen?
Allereerst wordt de lijst doorzocht op zoek naar het kleinste getal. In de array van getallen staat dit getal op een plaats die we even X noemen. Het kleinste getal vinden we op L(X). Het laagste getal wordt gewisseld met het eerste getal van de lijst: L(X) wordt gewisseld met L(1). Nu worden de getallen vanaf L(2) tot het einde L(E) doorzocht. De kleinste waarde die nu wordt gevonden, wordt in L(2) geplaatst. Daarna wordt gezocht vanaf L(3) tot en met het einde. Dit gaat net zo lang door totdat de lijst alleen nog maar wordt doorzocht vanaf het een na laatste getal L(E-1) tot en met het laatste getal L(E). Deze twee getallen worden vergeleken en eventueel gewisseld. De lijst is dan klaar.
Voor het wisselen van twee getallen wordt meestal en ook bij de Atari een *hulpvariabele* gebruikt.

```
LET Q = L(B)
LET L(B) = L(A)
LET L(A) = Q
```

Door deze drie regels wordt de waarde van de variabele L(B) tijdelijk opgeslagen in Q. Vervolgens wordt de waarde van de variabele L(A) gekopieerd in L(B). Daardoor is L(A) vrij (de oude waarde zit er nog wel in, maar deze is niet meer nodig). Aan de variabele L(A) wordt nu de waarde van Q toegekend. Daarin bevindt zich de oude waarde van L(B) en daardoor is de waarde van L(B) terecht gekomen in L(A) en de waarde van L(A) is terecht gekomen in L(B).
Deze regels verwisselen de inhoud van twee variabelen. Dit kan zowel bij arrayvariabelen als bij alle andere variabelsoorten gebruikt worden. Het is niet mogelijk de inhoud van twee ongelijksoortige variabelen te wisselen.
De hiervoor beschreven methode van het rangschikken van gegevens lijkt lastiger dan op het eerste gezicht nodig lijkt voor zoiets 'eenvoudigs' als sorteren. De computer begint met getal L(1) en vergelijkt dat met alle andere getallen. Telkens als een getal kleiner is, worden de twee getallen omgewisseld en zoekt de computer de lijst verder af, beginnend met het volgende getal Een programma zou er zo uit kunnen zien:

```
10 REM SORTEERPROGRAMMA
20 GRAPHICS 0
30 READ N
40 DIM L(N)
50 PRINT :PRINT "De datalijst: "
60 PRINT
70 FOR A=1 TO N
80 READ Z:L(A)=Z
90 PRINT L(A);" ";
100 NEXT A
110 REM
120 FOR X=1 TO N-1
130 FOR Y=X+1 TO N
140 IF L(Y)>=L(X) THEN GOTO 180
150 Q=L(Y)
160 L(Y)=L(X)
170 L(X)=Q
180 NEXT Y
190 NEXT X
200 PRINT :PRINT
210 PRINT "De gesorteerde lijst: "
220 PRINT
230 FOR A=1 TO N
240 PRINT L(A);" ";
250 NEXT A
300 DATA 9
310 DATA 12,3,109,16,7,88,8,-1,43,65
```

In regel 30 wordt het eerste gegeven gelezen. Dit gegeven hoort niet tot de te sorteren lijst maar dient *alleen* om het aantal gegevens in de datalijst op te geven. De regels 70, 80 en 100 lezen de gegevens in en plaatsen deze in de array L. De gegevens worden ter controle door regel 90 op het scherm afgedrukt. Het eigenlijke sorteren gebeurt in de regels 110-190. Telkens worden twee getallen vergeleken. Is het getal dat op een hogere plaats in de array staat groter dan het ermee vergeleken getal dan worden de twee getallen niet verwisseld. In dat geval springt de computer naar regel 150 waar de eerste van twee NEXT-opdrachten ervoor zorgt dat het volgende getallenpaar genomen wordt. Het resultaat wordt afgedrukt door de regels 170-250. In regel 300 staat het gegeven dat de lengte van de datalijst aangeeft. In regel 310 staat de datalijst. Wilt u stap voor stap zien hoe de computer de getallen telkens verwisselt dan kunt u de volgende regels toevoegen:

```
132 FOR A=1 TO N
133 POSITION 20,A*2+5
134 PRINT L(A);" ";
135 FOR Z=1 TO 50 : NEXT Z
136 NEXT A
137 FOR Z=1 TO 300 : NEXT Z
```

Zelfs met een zeer kleine vertraging om het wisselen zichtbaar te maken, kost het veel tijd voordat zo'n korte lijst met gegevens geordend is.
Het bovenstaande programma is een leuk sorteerprogramma, en het stukje waarin het sorteren daadwerkelijk gebeurt, kan goed als subroutine in andere programma's gebruikt worden. Het programma is echter niet echt efficient. De computer moet immers elke keer opnieuw de hele lijst met getallen doorwerken om te kijken of de twee getallen omgewisseld moeten worden of niet. Er bestaat een veel snellere wijze van sorteren. Dit wordt de **'bubble-sort'** genoemd. Bij deze manier van sorteren worden telkens de getallen die direkt naast elkaar staan (of bij een lijst met onder elkaar staande getallen, de onder elkaar staande getallen) vergeleken. De getallen worden gewisseld als de volgorde van grootte van de twee verkeerd is. De computer vergelijkt dan het hoogste van de twee getallen met het getal dat daar weer naast staat en wisselt deze eventueel om. Enzovoort.
Bij de eerste keer dat de computer door de lijst loopt, wordt L(1) vergeleken met L(2). Als L(2) kleiner is dan L(1) worden ze gewisseld. Daarna wordt L(2) waarin op dat moment het grootste van de twee getallen L(1) en L(2) zit, vergeleken met L(3). Ook nu wordt er gewisseld als L(3) kleiner is. Dit proces gaat zo door totdat L(E-1) vergeleken is met L(E) en deze eventueel gewisseld zijn.
Bij de tweede keer dat de computer door de lijst loopt worden op dezelfde manier de getallen van L(1) tot en met L(E-1) vergeleken. In L(E) zit nu immers het grootste getal van de vergelijkingen in de eerste ronde. Het proces wordt herhaald totdat uiteindelijk alleen L(1) en L(2) vergeleken worden. Op deze wijze worden de grootsten naar beneden gebracht en komen de kleine getallen vanzelf naar boven *bubbelen* (vandaar de naam).
Om te zorgen dat de computer niet elke keer de lijst blijft doorlopen terwijl de lijst eigenlijk al op volgorde staat, kan een extra controle ingebouwd worden. Elke keer als de computer met een nieuwe ronde begint, wordt er een geheugenplaatsje op 'klaar' gezet. Als de computer bij het doorlopen van de lijst een keer wisselt, wordt deze geheugenplaats op 'niet-klaar' gezet. Voordat de computer met een nieuwe ronde door de lijst begint, kijkt hij bij deze geheugenplaats of deze op 'klaar' of 'niet-klaar' staat. Staat de geheugenplaats op 'klaar' dan is er in de vorige ronde niets gewisseld en staan alle getallen op volgorde. Staat de geheugenplaats op 'niet-klaar' dan is er in de vorige ronde nog minstens een wisseling gemaakt en moet er nog een ronde komen. Het programma ziet er zo uit:

```
10 REM BUBBLESORT
20 GRAPHICS 0
30 READ N
40 DIM L(N):DIM K$(10)
50 PRINT :PRINT "De datalijst: "
60 PRINT
70 FOR A=1 TO N
80 READ Z:L(A)=Z
90 PRINT L(A);" ";
```

```
100 NEXT A
110 REM SORTEREN
120 LET E=N-1
130 LET K$="NIET-KLAAR"
140 REM BEGIN LUS
150 LET K$="KLAAR"
160 FOR G=1 TO E
170 IF L(G)<=L(G+1) THEN GOTO 210
180 REM WISSELEN
190 Q=L(G+1):L(G+1)=L(G):L(G)=Q
200 LET K$="NIET-KLAAR"
210 NEXT G
220 IF K$<>"KLAAR" THEN GOTO 150
230 PRINT :PRINT
240 PRINT :PRINT "De gesorteerde lijst: "
250 PRINT
260 FOR A=1 TO N
270 PRINT L(A);" ";
280 NEXT A
290 DATA 20
300 DATA 13,21,99,76,56,74,39,-1,111,-44
310 DATA 38,92,37,52,17,-1,-999,67,779,2
```

Het is natuurlijk korter om in plaats van de stringvariabele K$ met de waarde 'niet-klaar' of 'klaar' een numerieke variabele met de waarden 0 of 1 te nemen. In bovenstaand programma is *alleen* gebruik gemaakt van de woorden om u het gebruik van een zogenaamde 'vlag' duidelijk te maken. Een vlag is een teken dat aangeeft of een bepaalde situatie aanwezig is. Alleen als de vlag op 'klaar' staat moet opgehouden worden met het doorlopen van de lijst.
Het sorteren met behulp van de bubblesort is niet altijd de snelste. In het hobby-gebruik is echter de meest voorkomende situatie dat enkele gegevens toegevoegd worden aan een al bestaande en gesorteerde lijst. In deze gevallen is de bubblesort de snelste methode die er is.
Behalve het sorteren van getallen kunt u ook woorden op alphabetische volgorde laten zetten. U kunt in het hoofdstuk over strings vinden, wanneer de ene string groter is dan de andere. In het hoofdstuk over gegevenslijsten kunt u nalezen hoe u een bepaald deel van een string aanwijst.
Onderstaand vindt U een programma dat alles sorteert.

```
10 REM ALLESSORTEERDER
20 GRAPHICS 0
30 READ N
40 DIM N$((N+1)*20),C(N),P(N),J(N),A$(1),Q$(20),Z$(20)
50 N$((N+1)*20)=" ":N$(1,2)="  ":N$(2)=N$
60 FOR A=0 TO N-1
70 READ Z$:N$((20*A)+1,20*(A+1))=Z$
80 READ K,L,M:C(A)=K:P(A)=L:J(A)=M
85 PRINT N$
90 NEXT A
```

```
100 REM KEUZE MAKEN
110 PRINT "U kunt kiezen uit sorteren naar:"
120 PRINT "-PROGRAMMANAAM (N)"
130 PRINT "-CASSETTENUMMER (C)"
140 PRINT "-JAAR (J)":PRINT
150 PRINT "Uw keuze:";:INPUT A$
160 IF A$="N" OR A$="n" THEN GOTO 300
170 IF A$="C" OR A$="c" THEN GOTO 380
180 IF A$="J" OR A$="j" THEN GOTO 200
190 GOTO 100
200 E=N-1:K=0
210 REM JAAR
220 K=1
230 FOR G=0 TO E
240 IF J(G)<=J(G+1) THEN GOTO 260
250 GOSUB 800
260 NEXT G
270 E=E-1
280 IF K<>1 THEN GOTO 220
290 GOTO 500
300 E=N-1:K=0:REM PROGRAMMANAAM
310 K=1
320 FOR G=0 TO E
330 IF N$((20*G)+1,20*(G+1))<N$((20*(G+1))+1,20*(G+2)) THEN GOTO 350
340 GOSUB 800
350 NEXT G
360 E=E-1:IF K<>1 THEN GOTO 310
370 GOTO 500
380 E=N-1:K=0
390 REM CASSETTENUMMER
400 K=1
410 FOR G=0 TO E
420 IF C(G)<=C(G+1) THEN GOTO 440
430 GOSUB 800
440 NEXT G
450 E=E-1
460 IF K<>1 THEN GOTO 400
500 REM AFDRUKKEN
510 GRAPHICS 0
520 PRINT "De gesorteerde lijst is: "
530 POSITION 0,2:PRINT "PROGRAMMANAAM   CAS.  NR.   JAAR"
540 FOR A=0 TO N-1
550 POSITION 0,A+4:PRINT N$((20*A)+1,20*(A+1))
560 POSITION 20,A+4:PRINT C(A)
570 POSITION 25,A+4:PRINT P(A)
580 POSITION 30,A+4:PRINT J(A)
590 NEXT A
600 DATA 14
610 DATA SORTEER,1,100,1984
620 DATA FLIPPER,10,150,1985
630 DATA VLAKKEN,7,125,1984
640 DATA KUBUS,6,150,1984
650 DATA TEKSTVERW,12,75,1985
660 DATA RAKETJE,3,000,1983
670 DATA HELICOPTER,3,050,1983
680 DATA PALINDROOM,2,100,1984
690 DATA IJSSCHOTS,9,50,1985
700 DATA GEMIDDELDEN,3,150,1983
710 DATA TEST1,0,0,0
720 DATA TEST2,0,10,0
730 DATA TEST3,1,0,0
```

```
740 DATA TEST4,0,50,1
790 END
800 REM WISSELROUTINE
810 Q$=N$((20*G)+1,20*(G+1))
815 N$((20*G)+1,20*(G+1))=N$((20*(G+1))+1,20*(G+2))
820 N$((20*(G+1))+1,20*(G+2))=Q$
830 Q=C(G):C(G)=C(G+1):C(G+1)=Q
840 Q=P(G):P(G)=P(G+1):P(G+1)=Q
850 Q=J(G):J(G)=J(G+1):J(G+1)=Q
860 K=0
870 RETURN
```

12. BESTANDEN VERWERKEN

Het nut van bestanden

Zoals in de vorige hoofdstukken al bleek is het handig om een lijst van gegevens niet elke keer in een programma te hoeven opnemen. De meest efficiente methode is om aparte programma's en aparte lijsten met gegevens te hebben. Stel u heeft een hele serie met gegevensverwerkende programma's en een aantal lijsten met gegevens. Deze gegevenslijsten staan geheel los van de programma's. Op deze manier kunt u telkens een lijst met gegevens laten verwerken door een bepaald programma. Een paar standaard programma's zouden dan kunnen zijn:

- een programma om de gegevens in een grafiek weer te geven,
- een sorteerprogramma,
- een printprogramma (al dan niet met tabellen),
- een programma om nieuwe gegevens in te voeren en oude gegevens te verbeteren of te verwijderen,
- een statistisch programma,
- een spelprogramma of
- een leerprogramma.

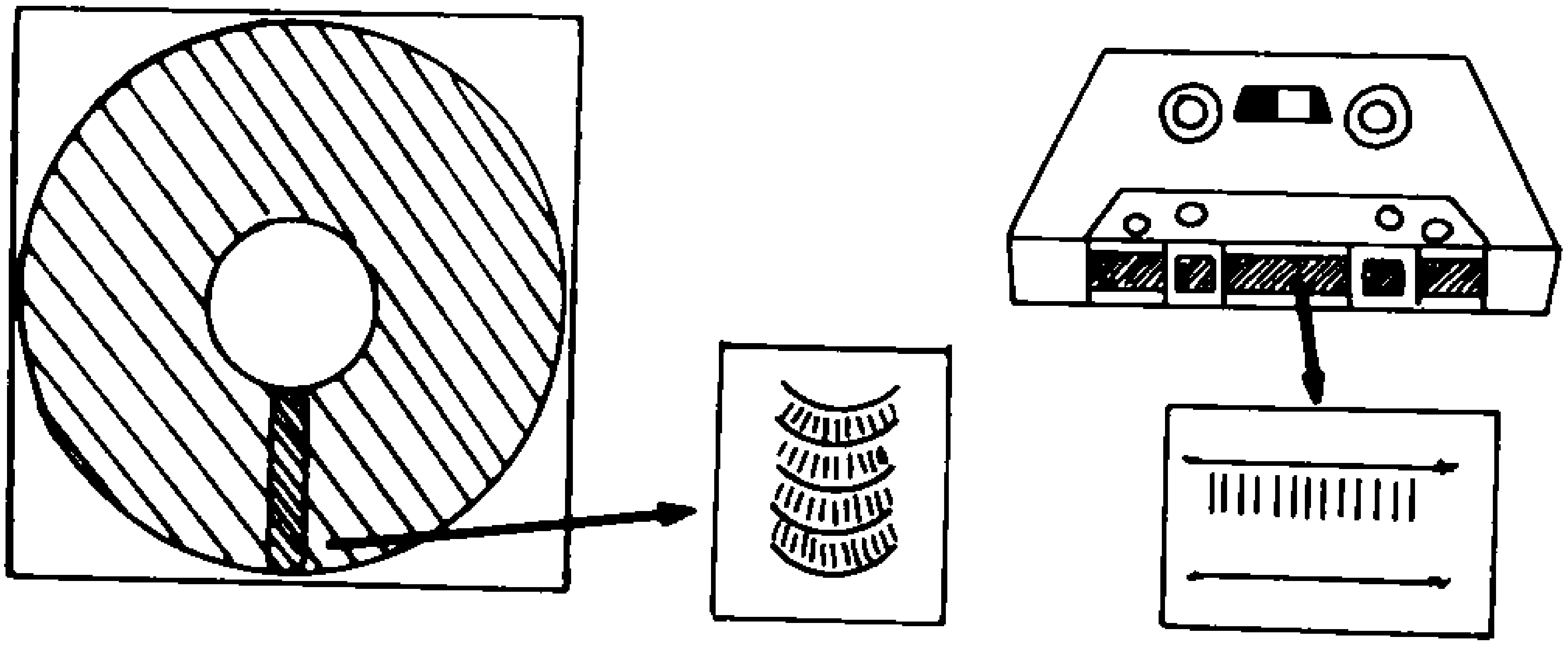
Niet ieder programma kan dezelfde soort gegevens gebruiken, maar er zijn wel heel veel overlappingen. Het is ook mogelijk verschillende funkties in een programma op te nemen, maar de nadelen hiervan zijn duidelijk: een lang programma, minder overzicht door een ingewikkelder programmastruktuur en minder geheugenruimte vrij voor de lijst met gegevens. Als mogelijke gegevensverzamelingen kunnen genoemd worden:

- energieverbruik,
- programmatheek,
- platen of cassette-discotheek,
- bibliotheek,
- onderzoekcijfers,
- woordenboeken,
- educatieve uitgaven,
- spelgegevens of
- tekengegevens.

De Atari beschikt over ruim voldoende faciliteiten voor het maken van aparte gegevensbestanden. Deze gegevensbestanden worden in het Engels 'files' genoemd. Dit woord zult u in de computerliteratuur veel tegenkomen. Hoewel een deel van de opdrachten, die de Atari gebruikt voor het werken met een bestand, onafhankelijk zijn van het opslagmedium (cassette of diskette) zijn er wel enkele verschillen bij het openen en sluiten van een bestand. De verschillende methodes zullen beide in dit hoofdstuk behandeld worden.

Het verschil tussen cassette- en diskbestanden.

Voor een goed begrip van het werken met bestanden is het handig iets af te weten van de manier waarop bestanden worden opgeslagen op cassette of diskette.
Elke verzameling van bij elkaar horende gegevens wordt een bestand (file) genoemd. De bestanden worden als magnetische signalen vastgelegd op magnetische dragers. In de hobbysfeer zullen dit voornamelijk cassettes of diskettes (kleine, soepele schijven al dan niet in een harde kunststof hoes) zijn.
Het grote verschil tussen een diskette en een cassette is dat de laatste een lange lijst van gegevens bevat, die gelezen moet worden door bij het begin van de lijst te beginnen en bij het einde van de lijst te eindigen. Dit is een gevolg van de wijze waarop signalen op een cassette worden gezet. Het bandje spoelt immers langzaam langs de koppen van de recorder en tijdens dit spoelen worden signalen op de cassette gezet. Bij het aflezen moet het bandje naar het begin worden terug gespoeld en gaat het aflezen ook weer van begin tot einde.
Een disk bestaat - eenvoudig voorgesteld - uit een heleboel cassettebandjes binnen elkaar, waarbij de bandjes in een cirkel zijn gelegd. Bij het 'einde' van zo'n cirkelvormig bandje (een track=spoor) gekomen is onmiddellijk weer 'het begin'.



schematische weergave van een diskette en een cassettebandje

De kop van een diskdrive (de machine waarmee disk gelezen en beschreven kunnen worden) kan zeer snel van het ene spoor naar het andere spoor springen. Door de cirkelvorm en de mogelijkheid om van het ene spoor naar het andere spoor te springen kan bij gebruik van disks direkt een bepaald gegeven uit een bepaald bestand worden gelezen. Het bestand hoeft niet van begin tot eind doorgelezen te worden. De diskette bevat naast de gegevens altijd een inhoudsopgave van de bestanden, waardoor de computer weet waar de kop naartoe moet springen.
Doordat een disk ook nog vele malen sneller draait dan een cassette spoelt, is de snelheid waarmee gewerkt kan worden bij disks vele malen hoger dan bij cassettes.

Het grote nadeel van disks en diskdrives is echter dat zij een stuk duurder zijn. Zolang snelheid niet een eerste vereiste is bij het gebruik van bestanden, is een cassetterecorder (bij Atari vaak een programmarecorder genoemd) een goede en goedkope oplossing.

Het maken van een bestand

Een bestand is te vergelijken met de lijsten van gegevens zoals die tot hiertoe telkens in DATA-opdrachten werden gezet. Er zijn echter twee grote verschillen:

- Gegevens in DATA-opdrachten kunnen alleen gebruikt worden in het programma waarin de gegevens staan. Gegevens uit een bestand kunnen door verschillende programma's worden gebruikt.
- Gegevens in DATA-opdrachten moeten door de gebruiker ingetikt worden, terwijl de gegevens in bestanden door de computer gemaakt kunnen zijn.

Er zijn veel methoden waarop het gebruik van bestanden aanschouwelijk gemaakt kan worden. Ongeacht de voorstellingswijze dienen de volgende punten in gedachte te worden gehouden:

- elk bestand heeft een eigen naam.
- een bestand bestaat uit een aantal geordende gegevens, waarbij elk gegeven een vaste plaats heeft binnen dat bestand.
- bestanden moeten voor gebruik worden geopend en na gebruik weer netjes **gesloten.**
- gegevens worden een voor een in een bestand gezet of uit het bestand gelezen (eigenlijk gekopieerd, want het origineel blijft in het bestand staan).

N.B.: in de meeste gevallen is de Atari zo vriendelijk voor ons het sluiten van het bestand over te nemen. U hoeft daar zelf bijna nooit op te letten. Bij andere merken computers is het sluiten van het bestand echter essentieel. Meestal wordt voor het sluiten van een bestand een vorm van de CLOSE (=sluit) opdracht gebruikt.

Het onderstaande programma geeft de gebruiker de mogelijkheid om een of meer getallen in te tikken. De computer schrijft de verzameling getallen weg naar een diskette.

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Doe DISKETTE in drive"
30 PRINT "Is de diskette geformateerd?"
40 PRINT "De diskette mag niet beschermd zijn "
50 PRINT "tegen schrijven!"
100 OPEN #7,8,0,"D:TEST.FIL"
110 PRINT :PRINT
120 PRINT "Tik een getal + RETURN";
130 INPUT GE
140 PUT #7,GE
150 PRINT
160 PRINT "Klaar? (j/n)";
170 INPUT A$
180 IF A$<>"J" THEN PRINT :GOTO 120
190 END
```

De eerste regel zorgt voor het dimensioneren van een string met een lengte een zodat de gebruiker op de vraag in regel 160 ('klaar?') met een 'J' van ja of een 'N' van nee kan antwoorden.
De regels 10 - 50 geven de gebruiker instrukties over het insteken van een diskette. De diskette moet geformateerd zijn en mag niet door middel van een sticker over de uitsparing, beschermd zijn tegen beschrijven. In dat geval kan de computer immers de gegevens niet kwijt.
De Atari heeft acht verschillende kanalen voor invoer en uitvoer van gegevens. Deze zijn gegroepeerd in blokken: Input/Output Control Blocks. In de literatuur over de Atari komt u die tegen als IOCB's. Twee van deze blokken wordt helemaal gebruikt voor het besturen van het scherm en het toetsenbord. Een blok wordt gebruikt voor gegevensuitwisseling met een diskdrive of een programmarecorder. De andere vijf zijn vrij voor eigen gebruik:

IOCB nr.	Gebruik
0	Scherm opmaak (E:)
1	Vrij gebruik
2	Vrij gebruik
3	Vrij gebruik
4	Vrij gebruik
5	Vrij gebruik
6	Grafische funkties. Opent en sluit automatisch
7	I/O naar cassette (C:) of diskette (D:). Opent en sluit automatisch

In regel 100 wordt kanaal 7 geopend. Dit klinkt vreemd. In de tabel hierboven staat immers dat dit kanaal automatisch geopend en gesloten wordt. Het automatisch openen en sluiten gebeurt echter alleen als de computer hiertoe

de noodzaak ziet doordat u een programma wilt wegzetten of inlezen. Door de opdracht LOAD, SAVE, LIST, enzovoort. Geeft u de computer al te kennen dat er iets gaat gebeuren met IOCB 7. In dit programma is er echter niets dat de computer wijst op het gebruik van IOCB 7, dus moeten we de computer expliciet meedelen dat we gegevens vanuit de computer willen wegsturen naar de diskdrive. De OPEN opdracht in regel 100 bestaat uit 4 gedeelten:

OPEN #kanaal, taak, 0, "D:xxx.xxx"

Het eerste getal dat u achter OPEN moet invullen is het IOCB nummer. Onmiddellijk voor dat nummer komt het amerikaanse aantal-symbool #. Het tweede gegeven dat u invult is een code voor de taak waarvoor het kanaal wordt geopend. Het maakt verschil voor de computer of u het IOCB kanaal nodig heeft om gegevens van de diskette in het geheugen van de computer te lezen, of dat u het kanaal nodig heeft om gegevens uit het computergeheugen op te slaan op een diskette of cassette. De taken met de erbij horende codes zijn:

Code	Taak
1	Verbeteren
2	Diskbewerking
4	Input
8	Output
16	Tekstvenster vlag ingesteld
32	Scherm niet wissen

De codes kunnen opgeteld worden om de verschillende taken te combineren. Zie ook de *bijlage IOCB*. In het programmavoorbeeld is gekozen voor code 8. Dat wil zeggen dat er gewoon output (uitvoer) van gegevens plaats zal vinden. Het derde gegeven, de nul, is bij diskbestanden altijd 0. Het vierde gegeven is een aanduiding voor de computer naar wat voor soort apparaat de gegevens gestuurd worden en onder welke naam de gegevens geplaatst gaan worden. De verschillende letters voor de apparaten zijn:

Letter	Apparaat
C:	Cassette- of programmarecorder
D:	Diskdrive (D1 t/m D4 bij gebruik van meer dan een disk-drive). Achter D: moet een bestandsnaam geplaatst worden.
E:	Scherm opmaak
K:	Keyboard (toetsenbord)
P:	Printer
S:	Beeldscherm
R:	RS-232 seriele uitgang

In het voorbeeld is de lettercode D: gebruikt omdat er gewerkt wordt met een diskdrive. Zie hieronder hetzelfde programma voor een programmarecorder. De naam achter de D: bepaalt hoe het bestand gaat heten. Als u met diskettes werkt moet elk bestand een eigen naam hebben. Dit is precies hetzelfde als het op een diskette opslaan van BASIC programma's. Die moeten ook allemaal een eenduidige naam hebben. Om de bestanden te kunnen onderscheiden van de BASIC programma's is het handig om gebruik te maken van de zogenaamde extension. Dit is de uitbreiding van de naam die aangeduid wordt door een punt met daarachter een soortnaam. In het voorbeeld is voor .FIL gekozen (van FILe = bestand). Als u op een diskette verschillende soorten gegevensopslag heeft is het handig om voor elke soort een aparte soortnaam te kiezen. Bijvoorbeeld:

.BAS	voor BASIC programma's
.FIL	voor gegevensbestanden
.TXT	voor tekstbestanden (van Atari-writer)

Gebruikt u veel gegevensbestanden, dan kunt u nog verder uitsplitsen. Bijvoorbeeld:

.GRA	voor gegevensbestanden met grafische coordinaten
.EDU	voor gegevensbestanden met educatieve gegevens
.NUM	voor gegevensbestanden met enkel numerieke gegevens enzovoort

We hebben nu in regel 100 een IOCB kanaal geopend naar de diskette. Wat gebeurt er nu? Zodra de computer bij dit deel van het programma komt, begint de diskdrive te zoemen en rommelen. Even gebeurt er schijnbaar niets en dan verschijnt de tekst "Tik een getal + RETURN". De computer is in het programma aanbeland bij regel 120. De computer heeft nu het kanaal geopend en op de diskette vast wat informatie opgeslagen die nodig is om later de diskette te kunnen lezen. Zo staat er nu op de inhoudsopgave van de diskette de naam van het nieuwe bestand (uit regel 100) en er staat waar dat bestand op de diskette begint. De kop van de diskdrive heeft zich daar naar toe begeven en wacht op de eerste signalen zodat die op de diskette geplaatst kunnen worden.
Dat die inhoudsopgave van de diskette erg belangrijk is kunt u merken door de computer te foppen. U laat het voorbeeldprogramma lopen zonder een diskette in de diskdrive te stoppen. U krijgt *foutmelding 144* te zien. Dat is de foutmelding die zegt dat u probeerde gegevens weg te schrijven op een diskette waarvan de uitsparing was beschermd door een *niet-schrijven* stickertje. De computer begrijp er niet veel meer van. Nog erger wordt het als u een diskette in de diskdrive doet die nog niet geformateerd is. Dan is er wel een diskette, maar de inhoudsopgave en de eerste essentiele gegevens van de diskette ontbreken. De computer slaat op hol. De diskdrive blijft draaien, u hoort af en toe een bliep, maar er gebeurt niets. Kortom: **zorg ervoor dat er een goede diskette in de diskdrive zit!**

De regels 110-130 vragen de gebruiker om een getal. Zodra u een getal invoert en afsluit met RETURN, krijgt de variabele GE (van gegeven) de waarde van het ingetikte getal. Regel 140 doet het eigenlijke werk.

PUT #7,GE

De PUT- (=plaats) opdracht plaatst het getal in het bestand dat geopend is via kanaal #7. Om geen foutmeldingen te krijgen moet u de PUT opdracht dus alleen gebruiken als u eerder in het programma een kanaal heeft geopend met het juiste getal (in het voorbeeld is dat gebeurd in regel 100).
Achter de PUT opdracht plaatst u een komma en de variabele waarvan de inhoud geplaatst moet worden in het bestand.
De regels 150-180 vragen de gebruiker of hij klaar is. Zo niet, dan wordt het programma teruggeleid naar regel 120, waardoor de gebruiker opnieuw om een getal wordt gevraagd. Dit getal wordt door de PUT opdracht in regel 140 direkt achter het vorige ingetikte getal in het bestand gezet.

Bestand met een programmarecorder

Ook met een cassetterecorder is het mogelijk een bestand te maken. Dit gaat in grote lijnen op dezelfde wijze als het maken van een diskettebestand. Leest u eventueel eerst het bovenstaande gedeelte door. Het volgende programma maakt een bestand op een cassette.

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Juiste cassette in recorder"
30 PRINT "Bij juiste tellerstand?"
40 PRINT "Druk na de dubbele toet de opname-"
50 PRINT "knop in en een willekeurige toets."
60 PRINT :PRINT "Druk RETURN als u klaar bent"
70 PRINT "met de voorbereidingen."
80 INPUT A$
90 OPEN #7,8,0,"C:"
100 PRINT :PRINT
110 PRINT "Tik een getal + RETURN";
120 INPUT GE
130 PUT #7,N
140 PRINT
150 PRINT "Klaar (j/n)?";
160 INPUT A$
170 IF A$<>"J" THEN PRINT :GOTO 110
180 END
```

De regels 5 - 80 zorgen voor de voorbereidende handelingen. De string om te antwoorden op vragen van het programma wordt in regel 5 gedimensioneerd.

Het juiste scherm wordt gekozen in regel 10 en de gebruiker krijgt allemaal aanwijzigingen op het scherm te zien.
U moet een leeg stuk cassette in de programmarecorder klaar zetten. Om later niet in de war te komen waar welk bestand staat, is het raadzaam voor elk bestand een aparte cassette (of cassettezijde) te nemen. Vindt u dit te prijzig worden dan kunt u ook wel verschillende bestanden op een cassette plaatsen, maar dan is het handig om elk bestand telkens bij een 'ronde' tellerstand (0000, 0100, 0200, 0300, 0400 etc.) te laten beginnen, en op een papiertje bij de cassette te noteren bij welke tellerstand, welk bestand begint. Wees vooral voorzichtig als u ook BASIC programma's op dezelfde cassette heeft staan, want voordat u het weet, heeft u per ongeluk vele uren tikwerk weg gewist.
Als de programmarecorder klaar staat met de cassette naar de juiste tellerstand gespoeld, kunt u volgens regel 60 en 70 op de RETURN toets drukken. Dit zorgt er alleen maar voor dat de computer wacht met verder gaan met het programma totdat u een teken gegeven heeft. Zodra u op RETURN drukt wordt de INPUT opdracht van regel 80 afgehandeld.
Nu wordt in regel 90 het IOCB kanaal geopend. Net als bij een diskettebestand wordt kanaal nummer 7 gebruikt. De taak waarvoor het kanaal geopend wordt is het weer wegschrijven van gegevens, dus is het tweede getal achter de OPEN opdracht **een 8.**
Het derde getal achter OPEN heeft bij het gebuik van cassettes een eigen betekenis. Het geeft de pauze tussen de verschillende 'blokken' met gegevens zoals die op de cassette komen aan. In de praktijk blijkt het het beste om hiervoor de waarde nul (0) te nemen. Een andere waarde zorgt weliswaar voor meer snelheid, maar ook voor meer kansen op fouten bij het wegschrijven.
Het vierde gegevens dat achter OPEN ingevuld moet worden is de lettercode van het gebruikte apparaat. In dit geval natuurlijk de C: van cassetterecorder.
Bij het gebruik van cassettes is het niet nodig om een naam te geven aan het bestand. Alle bestanden komen immers achter elkaar te staan (zie begin van dit hoofdstuk) of op aparte bandjes.

De regels 100 - 120 zorgen voor de invoer van een getal. Regel 120 zorgt voor het werkelijke wegzetten van de gegevens. De PUT opdracht gevolgd door het juiste kanaalnummer zorgt ervoor dat het gegeven op de cassette geplaatst wordt. Bij cassettebestanden worden de gegevens in groepen (zogenaamde blokken) van 128 op de cassette gezet. De computer spaart de gegevens op in een interne buffer (=opslagruimte) tot hij er 128 heeft, en de buffer geleegd moet worden, of totdat het programma beeindigd wordt, en de gegevens dus weggeschreven moeten worden voordat de buffer vol is.
De regels 140 - 170 vragen de gebruiker of hij/zij klaar is met het intikken van gegevens. Zo niet, dan moeten er nog meer gegevens in het bestand komen en wordt de computer in het programma terug gestuurd naar regel 110. Het gegeven dat nu wordt ingevoerd, wordt door de PUT opdracht in regel 130 direkt achter het vorige gegevens in het bestand geplaatst. Als de gebruiker voldoende gegevens ingetikt heeft, antwoordt hij met iets anders dan de 'J' van ja, en springt het programma naar regel 180.
De END opdracht van regel 180 zorgt ervoor dat het bestand op de cassette afgesloten wordt.

Lezen van een bestand

Met de bovenstaande programma's hebben we een bestand gemaakt op diskette of cassette. Om nu te kunnen controleren of de programma's precies gedaan hebben wat ze moesten doen, is het noodzakelijk dat we een gegevensbestand vanaf een diskette of cassette kunnen kopieren in het geheugen van de computer. Zodra de gegevens zich in het geheugen van de computer bevinden kunnen we ze op het beeldscherm plaatsen en zodoende controleren of de cijfers in het bestand gelijk zijn aan de eerder ingetikte cijfers. Ook het lezen van een bestand gaat met een diskdrive of een programmarecorder bijna gelijk. Het volgende programma leest de gegevens van het in het eerste programma van dit hoofdstuk gemaakte bestand, van de diskette.

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Doe DISKETTE in drive"
100 OPEN #7,4,0,"D:TEST.FIL"
110 PRINT :PRINT
120 PRINT "De getallen in het bestand zijn:"
140 GET #7,GE
150 PRINT GE;" ";
180 GOTO 140
190 END
```

De regels 5 - 20 zorgen weer voor wat diversen. Het dimensioneren van de antwoordstring, het juiste beeldscherm en een waarschuwing dat de juiste diskette in de drive moet zitten.
Heeft u een diskette die wel geformateerd is voor de Atari in de drive zitten, maar staat het bestand dat door het programma gelezen moet worden niet op de diskette dan krijgt u *foutmelding nummer 170* (bestand niet gevonden). Doet u een niet-geformateerde diskette in de drive, dan gaat de computer de fout in. Een tijd lang wordt geprobeerd om het bestand te vinden en uiteindelijk krijgt u *foutmelding 144.* De computer denkt dat u de schijf heeft afgeplakt.
Regel 100 opent het IOCB kanaal om het bestand te kunnen lezen.
Het kanaalnummer is weer 7.
Het taaknummer is dit keer 4. Het kanaal moet geopend worden voor INPUT (invoer) en niet voor uitvoer.
De nul is verplicht bij diskbewerkingen.
De D: staat voor diskdrive. De naam van het bestand moet gelijk zijn aan een bestaand bestand op de in de drive geplaatste diskette. Omdat we tot nu toe nog maar een bestand gemaakt hebben (in het eerste programma van dit hoofdstuk) is de naam gelijk aan het in dat programma gemaakte bestand. Let op de uitbreiding .FIL voor het aangeven van een bestand. De regels 110 en 120 zorgen voor een wat fraaiere weergave op het scherm. Regel 140 doet het werkelijke werk.

GET #7, GE

De GET (=neem) opdracht kopieert vanuit het bestand een gegeven van één byte lang, in het geheugen van de computer. De waarde die de GET opdracht vindt wordt geplaatst in de variabele die achter de komma van de GET opdracht staat. In dit geval de variabele GE.
Let op het kanaalnummer achter de GET opdracht. Dit dient gelijk te zijn aan het nummer van het kanaal dat eerder in het programma geopend is (in het voorbeeld is het kanaal geopend in regel 100).

Regel 150 drukt de waarde van de variabele GE af en maakt en spatie tussenruimte.
Regel 180 springt terug naar regel 140 om de volgende waarde uit het bestand te lezen. Als er geen gegevens meer zijn, sluit de computer zelf het kanaal en beeindigt door regel 190 het programma.

Als u dit programma laat lopen zult u een lijst te zien krijgen met getallen die u tijdens het lopen van het eerste programma in dit hoofdstuk heeft ingetikt. Het is mogelijk dat de getallen niet precies gelijk zijn aan de door u ingetikte getallen. Dat komt doordat de PUT en GET opdrachten de gegevens per byte (is acht bits) doorgeven. In een byte kan een getal dat niet groter is dan 255 worden opgeslagen. Voor een uitleg hierover kunt u de *bijlage G* bekijken. U kunt dit controleren door het eerste programma van dit hoofdstuk nogmaals te laten lopen en de getallen 256 en 257 in te tikken. Het bovenstaande programma zal als uitkomst geven dat de getallen 0 en 1 in het bestand staan.

Lezen van een cassettebestand

Ook het lezen van een cassettebestand gaat vergelijkbaar met het lezen van een diskettebestand. Ook nu weer moet de gebruiker wat extra moeite doen. Hij/zij dient ervoor te zorgen dat de cassette op de juiste tellerstand in de programmarecorder zit.

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Plaats cassette op juiste tellerstand"
30 PRINT "voor het gewenste bestand."
40 PRINT :PRINT "Na het klinken van de toeter"
50 PRINT "kunt u een willekeurige toets"
60 PRINT "behalve de BREAK toets, indrukken."
70 PRINT :PRINT "Druk op RETURN als u klaar bent."
80 INPUT A$
100 OPEN #7,4,0,"C:"
110 PRINT :PRINT
120 PRINT "De getallen in het bestand zijn:"
140 GET #7,GE
150 PRINT GE;" ";
```

```
180 GOTO 140
190 END
```

De regels 5 - 80 zorgen voor de inleidende huishouding. De gebruiker moet de cassette klaar zetten en op de RETURN-toets drukken als dat klaar is.
Regel 100 opent het IOCB-kanaal.
Het kanaalnummer is weer 7.
Het taaknummer is 4 voor invoer.
Het 'blokgetal' is nul. Dit is nu des te belangrijker omdat het bestand ook gemaakt is met bloknummer 0.
De lettercombinatie is C: van cassette.
De regels 110 en 120 zorgen voor een nette weergave.
Regel 140 doet het echte werk.
De GET opdracht neemt via kanaal 7 een gegeven van één byte lang en plaats dit gegeven in de variabele die achter de komma genoemd wordt (GE).
Zie ook hierboven bij de uitleg over diskettebestanden.
Regel 150 zorgt voor het afdrukken van de waarde van de variabele.
Regel 180 zorgt voor een lus die net zolang wordt doorlopen totdat er geen gegevens meer in het bestand staan.

Ook bij cassettebestanden geldt dat de GET en PUT opdrachten slechts een byte per keer behandelen, zodat de getallen maximaal 255 kunnen zijn (door speciale programmeertechnieken is daar wel iets op te vinden. maar het voert te ver om dat in dit boek te behandelen.
Tip: bedenk zelf een methode waarbij de computer kijkt of een getal 255 is en vervolgens het volgende getal bij dat getal betrekt. Samen zijn dan grotere getallen te bewerken.)

PRINT en INPUT

Naast de PUT- en GET-opdrachten kent de Atari ook de mogelijkheid van het maken van bestanden met behulp van de PRINT-opdracht. U kent de PRINT opdracht al van eerdere toepassingen, onder andere van het weergeven van een tekst op scherm, maar ook van het afdrukken van een tekst op een grafisch scherm door middel van PRINT #6. Dat ziet er bekend uit, niet? Door de PRINT #6 opdracht wordt een tekst afgedrukt via IOCB-kanaal #6. In het lijstje eerder in dit hoofdstuk, kunt u zien dat dit IOCB kanaal gereserveerd is voor grafische toepassingen.

Net als er via IOCB-kanaal #6 gePRINT kan worden, kunnen er ook gegevens via IOCB-kanaal #7 gePRINT worden in een bestand. Het volgende programma geeft een voorbeeld:

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Doe DISKETTE in drive"
30 PRINT "Is de diskette geformateerd?"
```

```
40 PRINT "De diskette mag niet beschermd zijn "
50 PRINT "tegen schrijven!"
100 OPEN #7,8,0,"D:GETTST.FIL"
110 PRINT :PRINT
120 PRINT "Tik een GETAL + RETURN";
130 INPUT GE
140 PRINT #7;GE
150 PRINT
160 PRINT "Klaar? (j/n)";
170 INPUT A$
180 IF A$<>"J" THEN PRINT :GOTO 120
190 END
```

Dit programma lijkt heel erg op het eerste programma in dit hoofdstuk. In regel 140 is echter de PUT-opdracht vervangen door de PRINT-opdracht. Let u wel op het gebruik van leestekens! Bij de PUT-opdracht werd een **komma** gebruikt, de PRINT-opdracht heeft een **punt-komma** nodig.

PRINT #7;GE

Ook in dit programma wordt het IOCB-kanaal geopend in regel 100 door een OPEN-opdracht. Kanaal #7 wordt gebruikt, de taak is **8** (uitvoer), de nul is verplicht bij diskdrives en de lettercode **D:** wordt gevolgd door een eenduidige naam met soortaanduiding achter de punt (GETTST.FIL).

Het voordeel van dit programma boven het eerste programma is, dat door de PRINT opdracht elk getal ingevoerd kan worden. Run dit programma en tik een paar getallen in. Schroom niet om de getallen 255, 256 en hoger in te tikken. In dit geval kan de computer ze wel aan. Heeft u de getallen ingetikt, run dan onderstaand programma.

```
5 DIM A$(1)
10 GRAPHICS 0
20 PRINT "Doe DISKETTE in drive"
100 OPEN #7,4,0,"D:GETTST.FIL"
110 PRINT :PRINT
120 PRINT "De getallen in het bestand zijn:"
140 INPUT #7,GE
150 PRINT GE
180 GOTO 140
190 END
```

Dit programma is de tegenhanger van het vorige. Met dit programma leest u de gegevens uit het bestand dat u met het vorige programma gemaakt heeft. Net als bij het computeren zonder bestanden is de tegenhanger van de PRINT-opdracht, de INPUT opdracht. In regel 70 ziet u dan ook de opdracht:

INPUT #7, GE

Tussen #7 en GE mag u *zowel* een punt-komma *als* een komma gebruiken. Zoals u ziet is dit programma overigens helemaal gelijk aan het vorige leesprogramma.
Als u het programma runt zult u zien dat de getallen groter dan 255, die u bij het runnen van het vorige programma ingetikt heeft, keurig in het bestand gezet zijn, en door dit programma goed uitgelezen worden.

Naast het juiste lezen van grote getallen, hebben de PRINT- en INPUT-opdrachten nog een tweede voordeel: het is mogelijk om ook strings te lezen. Daardoor kunnen ook teksten verwerkt worden met behulp van bestanden. Zie daarvoor de onderstaande twee programma's. (Kortheidshalve is de gebruikersuitleg sterk bekort.)
Het eerste programma maakt een bestand met strings.

```
5 DIM A$(1),GE$(15)
10 GRAPHICS 0
100 OPEN #7,8,0,"D:TEKSTST.FIL"
120 PRINT "Tik een TEKST + RETURN";
130 INPUT GE$
140 PRINT #7;GE$
160 PRINT :PRINT "Klaar? (j/n)";
170 INPUT A$
180 IF A$<>"J" THEN PRINT :GOTO 120
190 END
```

Het volgende programma leest het gemaakte bestand.

```
5 DIM A$(1),GE$(15)
10 GRAPHICS 0
20 PRINT "Doe DISKETTE in drive"
100 OPEN #7,4,0,"D:TEKSTST.FIL"
110 PRINT :PRINT
120 PRINT "De TEKSTEN in het bestand zijn:"
140 INPUT #7,GE$
150 PRINT GE$
180 GOTO 140
190 END
```

Zoals u ziet maakt het voor de computer weinig uit of er getallen of strings gelezen moeten worden. Het gaat allebei even goed. Het is zelfs mogelijk om zowel strings als numerieke gegevens door elkaar in een bestand op te slaan en later weer uit te lezen. Maar daarbij geldt hetzelfde als het door elkaar opslaan van strings en numerieke gegevens in een datalijst: ***U moet zelf zorgen dat een numeriek gegeven altijd aan een numerieke variabele wordt***

toegekend en dat een string gegeven aan een string variabele wordt toegekend.
Vergeet niet bij het werken met strings dat de variabele waar de strings in geplaatst moeten worden, altijd eerst gedimensioneerd moeten worden (zie in de programma's regel 5).

PRINT en INPUT bij cassettebestanden

Het werken met PRINT en INPUT bij het maken van cassettebestanden is vergelijkbaar met het werken met deze opdrachten bij diskettebestanden. De volgende twee programma's laten (zonder de gebruiksaanwijzingen) zien hoe u stringbestanden maakt. Voor de uitleg kunt u het gedeelte hierboven over diskettebestanden nalezen.

```
5 DIM A$(1),GE$(15)
10 GRAPHICS 0
100 OPEN #7,8,0,"C:"
120 PRINT "Tik een TEKST + RETURN";
130 INPUT GE$
140 PRINT #7;GE$
160 PRINT :PRINT "Klaar? (j/n)";
170 INPUT A$
180 IF A$<>"J" THEN PRINT :GOTO 120
190 END
```

Het volgende programma leest het bestand dat met het vorige programma gemaakt is. Vergeet niet de cassette op de juiste tellerstand terug te zetten voor het lezen.

```
5 DIM A$(1),GE$(15)
10 GRAPHICS 0
20 PRINT "CASS. OP JUISTE TELLERSTAND? (RETURN)"
30 INPUT A$
100 OPEN #7,4,0,"C:"
110 PRINT :PRINT
120 PRINT "De TEKSTEN in het bestand zijn:"
140 INPUT #7,GE$
150 PRINT GE$
180 GOTO 140
190 END
```

In alle programma's die de PRINT- en INPUT-opdracht maken, eindigt het programma dat het bestand leest met *foutmelding nummer 136:* EOF (End Of File – einde van het bestand). De computer geeft daarmee aan dat hij het einde van het bestand gevonden heeft. Een niet hinderlijk schoonheidsfoutje waar we in de komende programma's wat aan gaan doen.

Datalijsten en bestanden

De volgende programma's maken gebruik van een datalijst voor het invoeren van de gegevens die in het bestand moeten komen te staan. Dit lijkt een overbodige werkwijze. Waarom zou u alle gegevens in een bestand gaan plaatsen (hetgeen alleen maar tijd en ruimte op cassette of diskette kost) terwijl alle gegevens al ingetikt zijn.
Het voordeel dat blijft is, dat een éénmaal gemaakt bestand gebruikt kan worden door verschillende programma's. Een datalijst kan alleen gebruikt worden door het programma waar de datalijst onderdeel van is. Daarbij komt dat het nodig is dat de in te voeren getallen op de één of andere manier aan u getoond worden, anders kunt u ze niet overnemen. Een datalijst ziet er een beetje gestructureerder (en daardoor beter leesbaar) uit dan een opsomming van gegevens uit een bestand.
Het staat u natuurlijk vrij om de in het volgende programma gegeven getallen allemaal in een bestand te zetten volgens de methode van één van de vorige programma's, namelijk met een INPUT opdracht, direkt in te tikken. Maar eventuele fouten bij het intikken zijn dan slechts zeer lastig te verbeteren. Fouten in de datalijst kunt u verbeteren door de schermopmaaktoetsen te gebruiken en vervolgens het programma nog een keer te runnen.

Onderstaand programma geeft een voorbeeld hoe u gegevens eerst in een datalijst zet en deze lijst daarna door de computer in een dicht beschreven bestand laat zetten. De gebruikte gegevens zijn tekencoordinaten voor een aardrijkskundig programma. Het eerste gegeven in de aparte DATA-opdracht van regel 100 geeft alleen het aantal te gebruiken gegevens aan.

```
10 GRAPHICS 0
20 OPEN #7,8,0,"D:KAART.FIL"
25 PRINT "MOMENT A.U.B., BESTAND WORDT OPGEBOUWD"
30 READ A
40 FOR C=1 TO A
50 READ X,Y
60 PRINT #7;X:PRINT #7;Y
70 NEXT C
80 END
100 DATA 260
110 DATA 75,37,70,63,64,83,57,97,53,103,49
115 DATA 104,50,106,52,107,54,110,59,114
120 DATA 59,114,62,115,65,116,68,115,72,117
125 DATA 72,119,67,119,63,123,56,123,56,127,59,136
130 DATA 56,137,50,135,44,126,40,125,32,127
135 DATA 26,130,26,134,31,135,32,137,35,137
140 DATA 38,141,42,141,45,139,46,138,49,138
145 DATA 59,141,62,144,66,144,67,143,66,137
150 DATA 70,134,70,139,74,138,76,135,81,135
155 DATA 82,136,79,139,80,141,87,139,89,134
```

```
160 DATA 92,133,94,135,93,143,100,148,108,148
165 DATA 112,147,114,152,123,152,124,155,120,163
170 DATA 120,171,117,177,117,179,120,181,120,183
175 DATA 123,183,126,182,131,183,132,181,132,177
180 DATA 136,175,136,171,132,167,130,167,128,161
185 DATA 132,161,132,157,135,153,134,151,132,151
190 DATA 132,147,139,139,139,131,134,122,128,115
195 DATA 125,114,125,110,135,109,135,107,140,106
200 DATA 147,111,158,103,160,104,163,97,161
205 DATA 94,156,95,156,92,160,91,171,83,172,77
210 DATA 170,71,168,67,164,68,160,67,155,62
215 DATA 157,59,156,56,158,53,167,52,168,39
220 DATA 175,31,176,19,177,17,176,15,168,13
225 DATA 167,11,163,10,161,3,158,1,146,2
230 DATA 138,8,136,5,128,6,114,11,107,10,101
235 DATA 23,101,27,104,31,100,41,100,43
240 DATA 110,44,116,42,117,44,112,47,111,51
245 DATA 114,54,116,55,126,53,128,54,129,56
250 DATA 126,59,125,67,122,71,116,76,115,81
255 DATA 110,83,106,83,96,80,92,77,86,76
260 DATA 91,73,93,69,93,65,94,63,94,64,93
265 DATA 65,90,63,89,57,91,56,94,57
270 DATA 96,54,99,53,100,51,94,49,91,51,90
275 DATA 49,91,41,88,37,85,38,84,39
280 DATA 83,41,81,40,78,37,78,35,76,35,75
285 DATA 37,0,0,64,119,62,121,57,119
290 DATA 54,120,50,117,50,114,48,113,44,114
295 DATA 43,112,44,111,49,111,51,113,64,119
300 DATA 0,0,51,122,50,124,48,123,45,121
305 DATA 41,120,38,121,36,118,42,115,50,119
310 DATA 51,122,0,0,54,125,55,129,54,132
315 DATA 49,129,48,127,54,125,0,0,58,144
320 DATA 44,155,40,156,39,151,34,149,26,153
325 DATA 22,151,20,146,23,141,30,141,40,145
330 DATA 46,143,47,141,50,140,58,144,0
335 DATA 0,96,73,95,77,102,78,104,81,110,81
340 DATA 111,76,121,69,124,65,123,59,120
345 DATA 59,115,57,113,58,96,73,0,0,76,27
350 DATA 75,33,77,34,80,31,82,30,83,25
355 DATA 82,23,81,22,76,27,0,0,88,11
360 DATA 83,16,82,21,85,19,86,17,90,13
365 DATA 88,11,0,0,111,2,96,5,92,8
370 DATA 92,11,95,12,97,9,100,8,102,6,110,4,111,2
```

Regel 10 kent u: wissen van het scherm. Regel 20 is het openen van een IOCB kanaal. Nummer 7, taak 8 (uitvoer), de verplichte 0, de D: van diskette en een

toepasselijke naam. Omdat het om aardrijkskundige gegevens gaat, is dit bestand KAART.FIL genoemd.
Regel 25 zorgt voor een tekstje op het scherm terwijl de gegevens in het bestand gezet worden. Daardoor hoeft de gebruiker niet zo lang tegen een leeg scherm aan te kijken. In plaats van een enkele regel tekst, kunt u natuurlijk het hele scherm vol met prachtige tekeningen maken. Daarover meer in de hoofdstukken over grafiek.
In regel 30 wordt gelezen hoelang de datalijst is.
In regel 40 wordt de lus begonnen.
Regel 50 zorgt telkens voor het lezen van twee waarden: een X- en een Y-coordinaat.
Regel 60 zorgt ervoor dat de gegevens in het bestand geplaatst worden. Telkens worden achtereenvolgens de waarde van X en de waarde van Y via kanaal #7 naar het bestand gestuurd.
Regel 70 sluit de lus en regel 80 sluit het programma.
De regels 100 en verder vormen de gegevens die nodig zijn voor het bestand.

De gegevens die nu in het bestand staan, kunnen met het volgende programma weer gelezen worden.

```
10 GRAPHICS 0
20 OPEN #7,4,0,"D:KAART.FIL"
30 TRAP 100
40 INPUT #7,X,Y
50 PRINT X,Y
60 GOTO 40
100 PRINT "EINDE BESTAND"
```

In regel 20 het openen van het bestand. Nu met taaknummer 4 voor het lezen. Uiteraard moet de naam van het bestand gelijk zijn aan de naam van het zojuist gemaakte bestand.
Regel 30 wordt hieronder behandeld.
Regel 40 leest telkens twee waarden uit het bestand. De X-coordinaat en de Y-coordinaat.
Regel 50 drukt deze waarde af.
Regel 60 zorgt ervoor dat er telkens opnieuw een stel waarden gelezen wordt. Net zo lang totdat het einde van bestand bereikt is en de computer *foutmelding 136* geeft.
Dit laatste gebeurt echter niet door de opdracht in regel 30:
Regel 30 heeft een nieuwe opdracht:

TRAP (= val)

Deze opdracht zorgt ervoor dat de computer in een val loopt als hij een foutmelding wil geven. Eerder in dit hoofdstuk las u hoe de computer bij de INPUT opdracht, aan het einde van het bestand, een foutmelding geeft, ten teken dat het einde van het bestand gevonden is. Deze melding is reuze handig, maar komt niet altijd van pas.

Om de foutmelding te snel af te zijn, gebruikt u de TRAP-opdracht vlak voor de regel waar eventueel een foutmelding kan ontstaan. In dit geval kan de foutmelding ontstaan door de INPUT-opdracht van regel 40, dus moet de TRAP-opdracht op regel 30 staan.
De TRAP-opdracht moet een regelnummer hebben waar de computer naar toe zal springen als er een foutmelding gegenereerd wordt. In het voorbeeld is gekozen voor regel 100. Daar zorgt een PRINT opdracht voor het afdrukken van de tekst: EINDE BESTAND.
Dit is al een heel wat vriendelijker gezicht dan de foutmelding: ERROR 136.
U hoeft het niet te laten bij een enkele PRINT opdracht. U kunt de TRAP opdracht naar een complete subroutine laten springen.

Stel u heeft een programma dat de gebruiker telkens het omgekeerde van een getal geeft (dat wil zeggen: het omgekeerde van 10 is een-tiende = 0.1, het omgekeerde van 3 is een-derde = 0.333333 enzovoort).

```
10 GRAPHICS 0
20 PRINT "VOER GETAL IN, + RETURN";
30 INPUT C
50 PRINT :PRINT C,"OMGEKEERD: ";1/C
60 PRINT :PRINT
70 GOTO 20
```

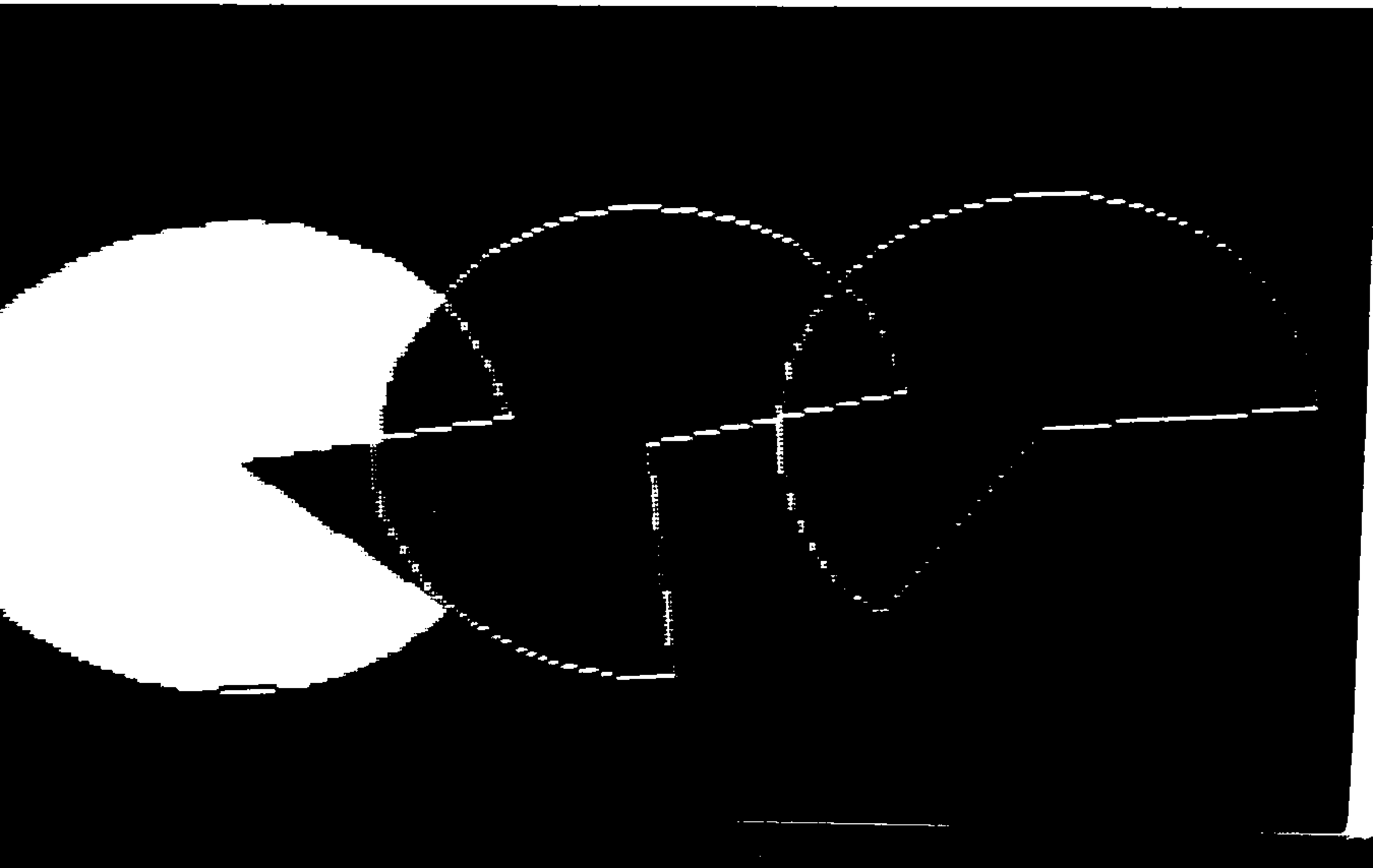
Dit programma laat u telkens een getal invoeren en deelt vervolgens 1 door dat getal.
Nu weet iedereen dat je niet door nul kunt delen. 1/0 bestaat niet, want er is geen enkel getal dat als je het vermenigvuldigt met 0, het resultaat 1 oplevert.
Als u in het bovenstaande programma een 0 invoert, krijgt u een foutmelding. Het programma is vervolgens afgelopen. Dat is jammer, want misschien was het wel gewoon een vergissing en had de gebruiker 10 willen intikken.
Door het programma te veranderen met een TRAP opdracht en een daarbij horende subroutine, blijft het programma keurig doorwerken, ook als de gebruiker een 0 intikt.

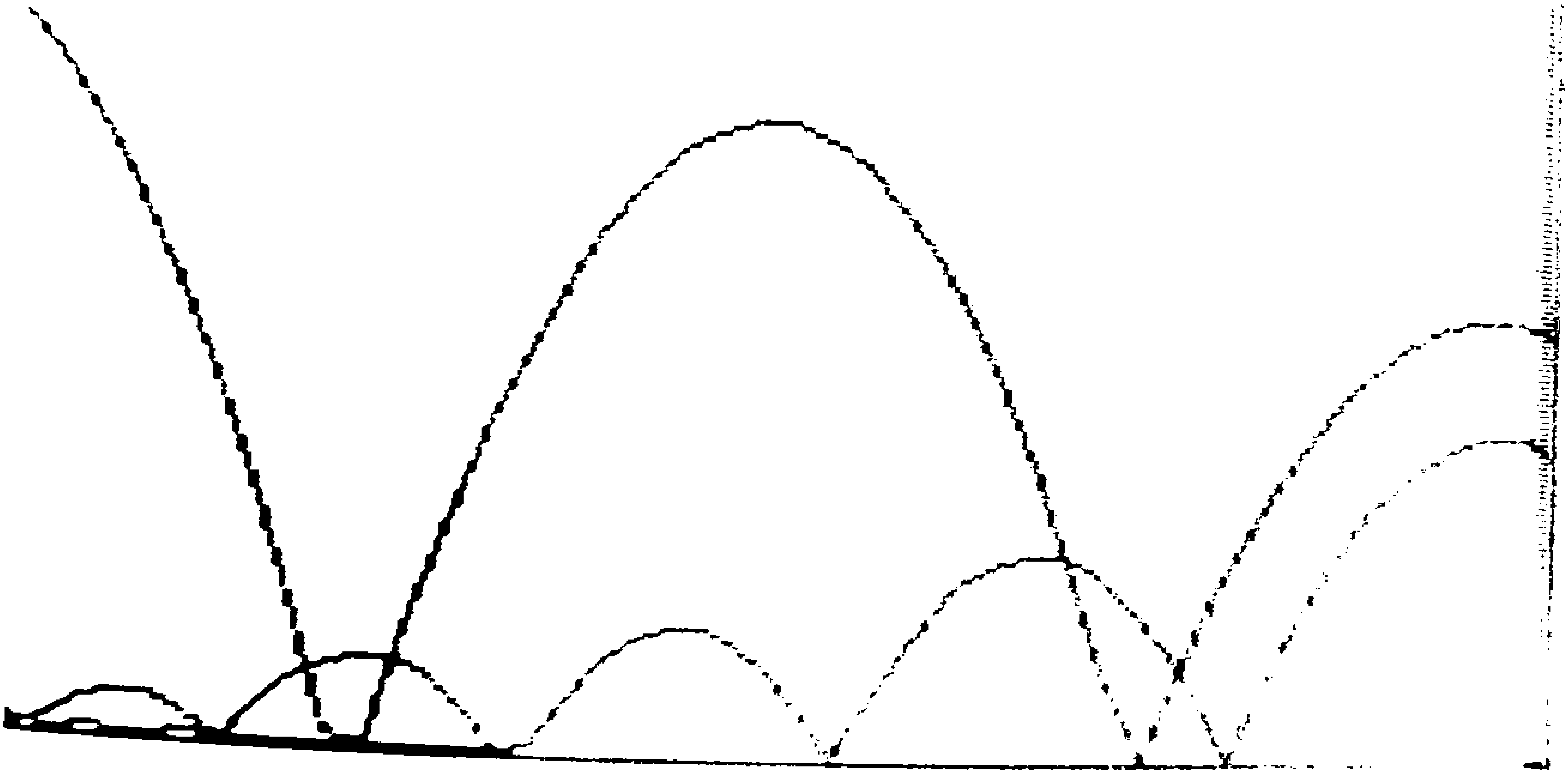
```
10 GRAPHICS 0
20 PRINT "VOER GETAL IN, + RETURN";
30 INPUT C
40 TRAP 100
50 PRINT :PRINT C,"OMGEKEERD: ";1/C
60 PRINT :PRINT
70 GOTO 20
100 PRINT :PRINT "Delen door 0 is niet toegestaan"
110 PRINT
120 GOTO 20
```

Terug naar het bestandverwerkende programma. De lijst van getallen die we kregen door het bestand te lezen, was wel indrukwekkend, maar niet wat de uiteindelijke bedoeling was; namelijk een aardrijkskundig programma. Het programma moet daarvoor uitgebreid worden met enkele tekenopdrachten.

Landschap met ruimteschip programma *blz. 205*

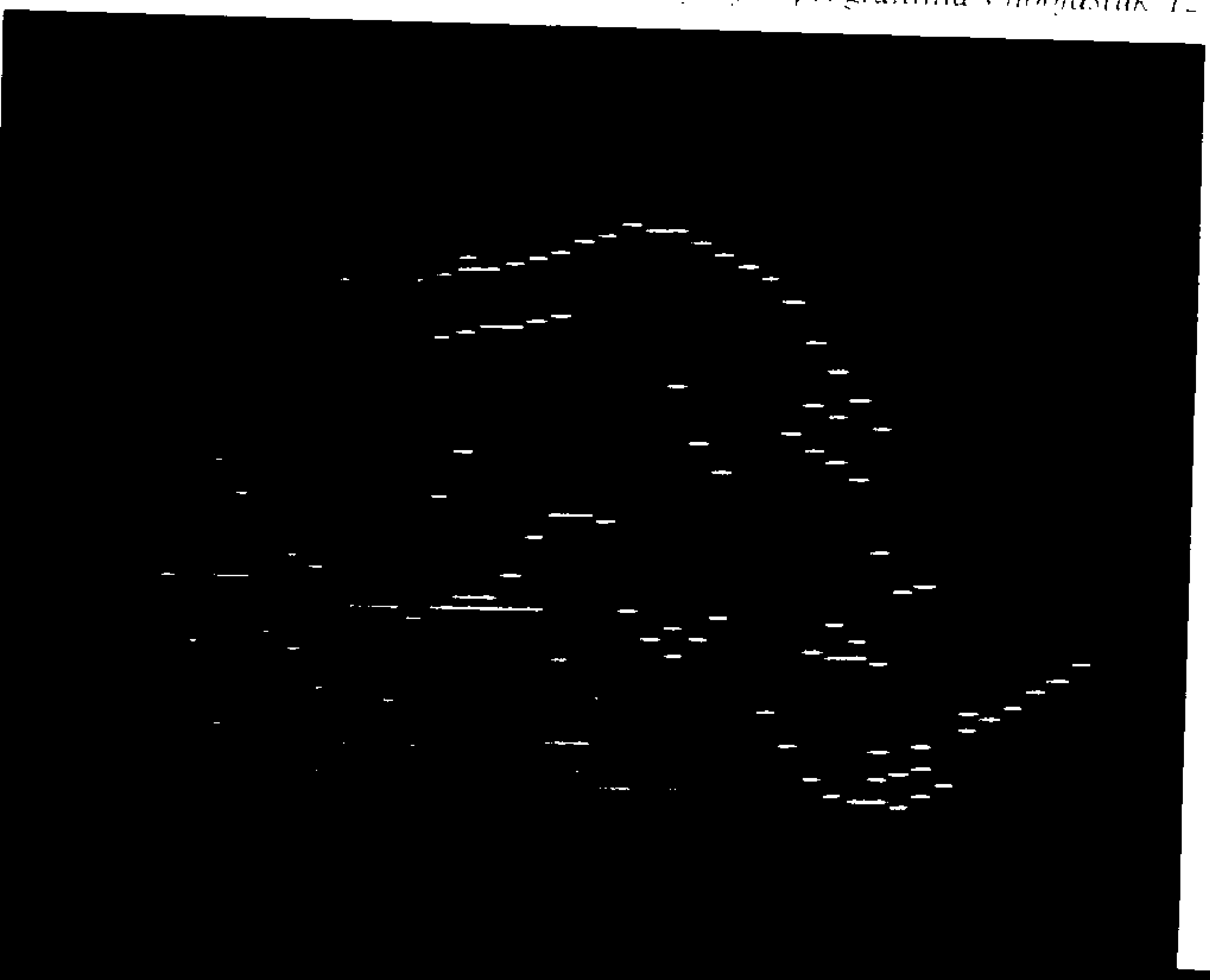
Schijven voor soortgelijke programma's [illegible]





Stuiterbal programma *blz. 20*[illegible]

Neushoorn [illegible] soortgelijke programma's *hoofdstuk 12*



Hierdoor ziet u gelijk dat een ander programma (het onderstaande) ook gebruik kan maken van hetzelfde bestand.

```
10 GRAPHICS 8+16
15 COLOR 1
20 PLOT 75,37
30 OPEN #7,4,0,"D:KAART.FIL"
40 TRAP 100
50 INPUT #7,X,Y
60 IF X>0 AND Y>0 THEN DRAWTO X,Y:GOTO 50
70 INPUT #7,X,Y:PLOT X,Y
80 GOTO 50
100 GOTO 100
```

In regel 10 wordt een grafisch scherm gemaakt. zonder tekstvenster (zie de hoofdstukken over grafiek).
In regel 15 wordt een kleur gekozen.
In regel 20 wordt de grafische cursor (vergelijkbaar met de tekstcursor. maar dan voor tekeningen) naar een bepaalde positie gedirigeerd. In regel 30 wordt het bestand via IOCB kanaal #7, geopend voor het lezen van gegevens.
In regel 40 wordt de foutmeldingsval opgezet. Vooral bij grafische programma's is dat belangrijk, omdat de Atari bij het vinden van een fout die gemeld moet worden, het grafische scherm weg haalt en het normale tekstscherm weer laat zien. Daarmee is de grafische afbeelding verdwenen. Regel 50 leest telkens twee waarden: een X- en een Y-coordinaat.
Regel 60 bekijkt of er naar het volgende punt, aangegeven door de coordinaten, een lijn getrokken moet worden. Nederland is een land met verschillende eilanden, en het is minder fraai al die eilanden met het vaste land te verbinden door een lijn. Zodra de computer begint aan het tekenen van een eiland, zijn de coordinaten (0,0). Dat is het teken voor regel 60 om naar dat punt geen lijn te trekken.
In plaats daarvan wordt regel 70 uitgevoerd waar staat dat het volgende coordinatenstel gelezen moet worden en dat de grafische cursor naar dat punt verplaatst moet worden, zonder een lijn te trekken.
Regel 50 zorgt voor een lus.
Regel 100 blokkeert het programma, zodat het grafische scherm zichtbaar blijft. De Atari verwijdert na afloop van een programma, het grafische venster. Wilt u dat nog een tijdje zien, dan moet u het programma met een kunstgreep kunstmatig aan de gang houden.
Om het programma te verlaten drukt u op BREAK of op RESET.

Wat is de kaart van Nederland zonder plaatsen? Gebruik het volgende programma om nog een tweede bestand te maken met daarin alle plaatsen.

```
5 DIM ST$(5)
10 GRAPHICS 0
20 OPEN #7,8,0,"D:STAD.FIL"
```

```
25 PRINT "MOMENT A.U.B., BESTAND WORDT OPGEBOUWD"
30 READ A
40 FOR C=1 TO A
50 READ X,Y,ST$
60 PRINT #7;X:PRINT #7;Y:PRINT #7;ST$
70 NEXT C
80 END
100 DATA 19
110 DATA 119,74,MAAST,84,76,AMSTE,72,74,HAARL,62,90,DEN H
120 DATA 68,105,ROTTE,78,124,BREDA,101,116,DEN B,98,90,UTREC
130 DATA 128,99,ARNHE,158,79,HENGE,117,22,LEEUW,147,15,GRONI
140 DATA 106,64,LELYS,150,34,ASSEN,68,85,LEIDE,77,36,DEN H
150 DATA 122,111,NIJME,109,135,EINDH,130,172,HEERL
```

Zoals u ziet heeft het bestand een eigen naam (STAD.FIL) en worden van alle coordinaten tevens aangegeven voor welke plaats zij dienen. Dit laatste is niet strikt noodzakelijk, maar maakt het gemakkelijker eventuele fouten op te sporen, of later het programma nog eens na te lezen.

Met het volgende programma kunt u beide bestanden lezen en verwerken.

```
5 DIM ST$(5)
10 GRAPHICS 8+16
15 COLOR 1
20 PLOT 75,37
30 OPEN #7,4,0,"D:KAART.FIL"
40 TRAP 100
50 INPUT #7,X,Y
60 IF X>0 AND Y>0 THEN DRAWTO X,Y:GOTO 50
70 INPUT #7,X,Y:PLOT X,Y
80 GOTO 50
100 OPEN #7,4,0,"D:STAD.FIL"
110 TRAP 200
120 INPUT #7,X,Y,ST$
130 PLOT X,Y:DRAWTO X+5,Y:DRAWTO X+5,Y+5
135 DRAWTO X,Y+5:DRAWTO X,Y
140 GOTO 120
200 GOTO 200
```

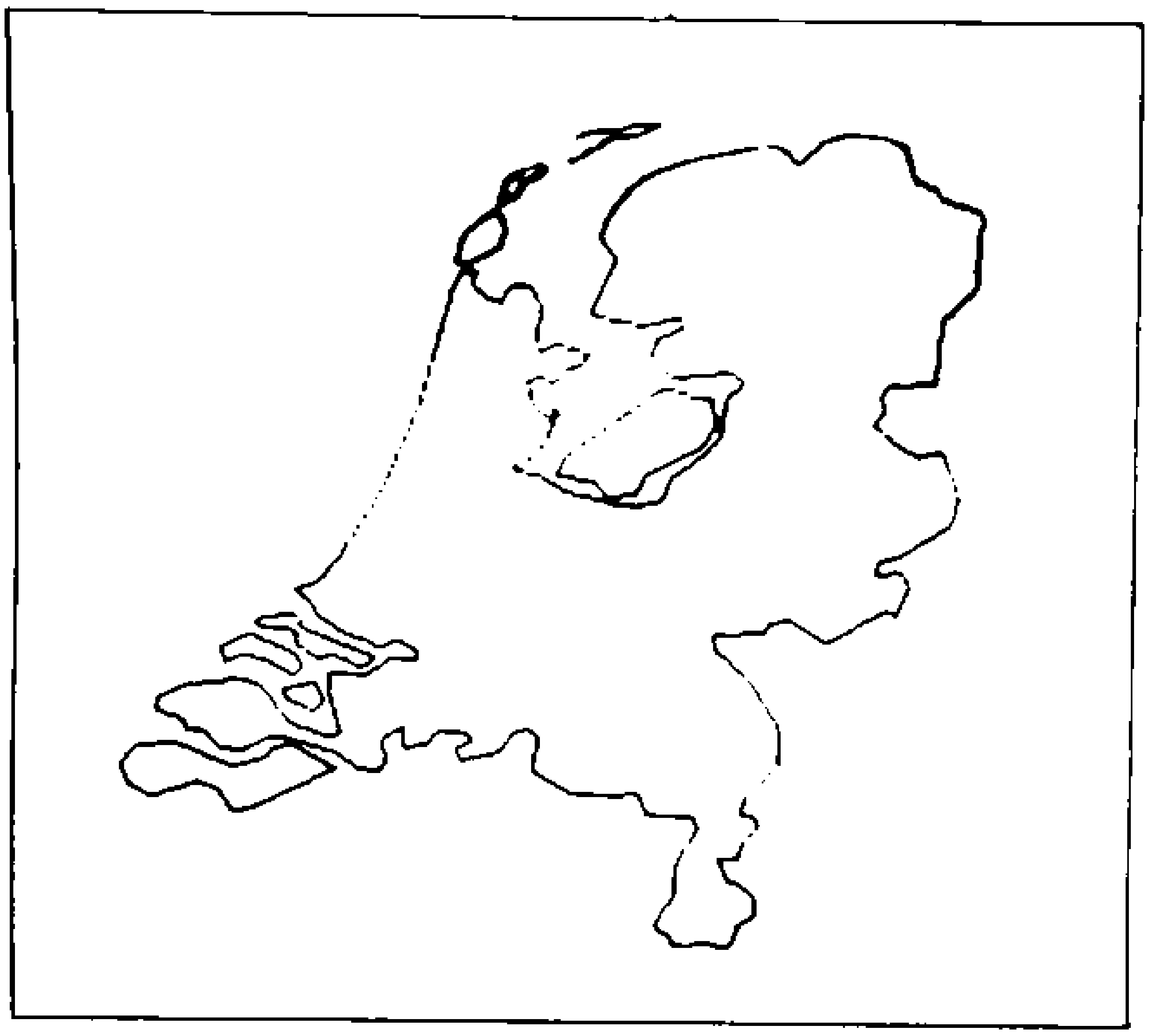
Let op het gebruik van de TRAP-opdracht in regel 40. Door deze opdracht wordt het programma na het beeindigen van het eerste bestand door verwezen naar het volgende bestand.

Cassettebestanden

Op analoge wijze aan de eerste programma's van dit hoofdstuk, kunnen de ge-

bruikers van een programmarecorder de programma's zodanig aanpassen dat zij ook met de recorder werken. Belangrijke punten daarbij zijn:

- Zorg dat de gebruiker de tijd krijgt de cassette op de juiste tellerstand in de recorder te plaatsen. Laat eventueel het programma door middel van een INPUT-opdracht wachten totdat de gebruiker klaar is.
- Gebruik de lettercombinatie C: in plaats van D:
- Een naam voor het bestand is niet nodig.
- Vermeld op het papiertje bij de cassette, bij welke tellerstand, welk bestand begint.



13. Uitstapjes

Inleiding

Tot op heden waren de in dit boek gebruikte programma's niet zo erg lang en waren er niet zo veel vertakkingen. Eens komt echter het moment, dat voor een bepaald probleem een kort programma niet meer voldoet. De computer moet zoveel opdrachten krijgen dat daar lange programma's voor nodig zijn. Gaat zo'n lang programma gepaard met veel vertakkingen dan wordt het programma steeds slechter te lezen. Afgezien daarvan is ook het maken van zo'n lang programma lastig, omdat de programmeur snel in de war kan raken en niet meer weet waar hij/zij is.
Om zo'n probleem aan te pakken kan men gebruik maken van **'differentiatie'** en **'specialisatie'**. Ofwel: het werk wordt verdeeld (differentiatie) en de werkers leggen zich toe op een bepaalde bewerking (specialisatie).
Dit kan ook bij het programmeren gedaan worden. Door bepaalde delen van het programma gespecialiseerde taken te laten volbrengen wordt het hoofdprogramma korter en beter leesbaar. In de ideale situatie is het hoofdprogramma niet meer dan een opzichter, die zelf als enige taak de verdeling en de coordinatie van het werk heeft, dat door andere delen van het programma wordt gedaan.

Subroutines

BASIC kent voor het laten uitvoeren van een bepaalde taak door een bepaald programmaonderdeel de subroutine. Het woord legt al iets uit. *'Sub'* duidt op een ander niveau van deelwerkzaamheden en *'routine'* duidt op het veelvuldig door dit programma-onderdeel uitvoeren van dezelfde taak.
Als voorbeeld laten we het laten uitrekenen van de grootste gemene deler van twee getallen door de computer zien. De taken die de computer achter elkaar moet uitvoeren zijn:

- weergeven van de instrukties
- invoer van de getallen
- getallen op volgorde van grootte zetten
- rekenen
- afdrukken van het resultaat

Deze deeltaken lijken wel erg klein voor het maken van aparte vertakkingen in een programma. Maar alleen door een klein overzichtelijk takenpakket als voorbeeld te nemen, kan de subroutine goed duidelijk gemaakt worden.
Als alle aparte taken door een strikte taakscheiding door aparte subroutines gedaan worden, komt het programma er als volgt uit te zien:

```
10 REM GROOTSTE GEMENE DELER
20 GOSUB 200
30 PRINT "UW EERSTE GETAL ";:INPUT E
40 PRINT "UW TWEEDE GETAL ";:INPUT T
50 GOSUB 300
60 GOSUB 400
70 GOSUB 500
80 END
200 REM INSTRUKTIES
210 GRAPHICS 0
220 PRINT "Dit prog. gebruikt de"
230 PRINT "Euclidische methode om de "
240 PRINT "grootste gemene deler van twee"
250 PRINT "getallen te vinden."
260 PRINT :PRINT :PRINT
270 RETURN
300 REM GROOTSTE GETAL IN A
310 IF E<T THEN Q=E:E=T:T=Q
320 RETURN
400 REM HET REKENEN
410 A=E:B=T
420 FOR Z=1 TO A
430 IF A-(Z*B)>=0 THEN NEXT Z
440 RE=A-((Z-1)*B)
450 A=B
460 B=RE
470 IF RE<>0 THEN GOTO 420
480 RETURN
500 REM AFDRUKKEN VAN RESULTAAT
510 GRAPHICS 0
515 PRINT :PRINT
520 PRINT :PRINT "    ****************************"
530 PRINT :PRINT "      DE GROOTSTE GEMENE DELER"
540 PRINT :PRINT "      VAN ";E;" EN ";T;" IS ";A
550 PRINT :PRINT "      ************************"
560 RETURN
```

Het te strikt toepassen van subroutines blijkt ook tot ongewenste resultaten te leiden. Hoewel het hoofdprogramma zeer kort is (regel 10-80), is dit geheel onleesbaar. De lezer van het programma moet telkens de subroutines bekijken om te zien wat het programma doet. Het is daarom aan te raden de ***REM-opdrachten*** die de werking van een subroutine verduidelijken onmiddellijk achter de verwijzing naar de subroutine (GOSUB) te plaatsen.
Let u op de speciale deling in de regels 420 - 440.
Dit is een deling van gehele getallen met een rest. Zo is 17/4 gelijk aan 4.25. Maar 17 op deze speciale manier gedeeld door 4, is gelijk aan 4 met een rest van 1. Bij enkele andere computers wordt dit aangegeven met MOD (van

modulo). U kunt dit voorbeeld dan tegen komen als 17 MOD 4 (dat is 4, rest 1).

Subroutines worden aangeroepen door de opdracht

GOSUB (ga naar subroutine)

Deze opdracht wordt gevolgd door het regelnummer waar de subroutine begint. Het is handig om daar hoge regelnummers voor te kiezen, zodat er altijd voldoende ruimte overblijft voor een ononderbroken hoofdprogramma. Zorg er voor dat de regel waar naar toe gesprongen wordt, wel bestaat! Anders geeft de computer foutmelding nummer 12: het regelnummer werd niet gevonden.
Let op! U mag bij GOSUB zowel regelnummers in de vorm van gehele getallen, als regelnummers in de vorm van numerieke berekeningen gebruiken. Zo kunt u bijvoorbeeld een nieuwe regel 5 toevoegen en regel 50 veranderen:

```
5 LIJN = 300
50 GOSUB LIJN
```

Om van de subroutine weer terug te keren naar het hoofdprogramma, dient de opdracht:

RETURN (keer terug)

Deze opdracht zorgt ervoor dat de computer terugspringt naar de regel volgend op de regel waar de GOSUB stond die bij deze RETURN hoort. In regel 20 staat GOSUB 200. De RETURN in regel 270 zorgt ervoor dat de computer terug springt naar de regel volgend op regel 20 en dat is in dit programma regel 30.
De opdracht

END (einde)

in regel 80 zorgt ervoor dat het programma stopt. Zou het programma aan het einde van het hoofdprogramma niet gestopt worden, dan zou de computer beginnen met de regels vanaf 200. Maar deze regels zijn bedoeld als subroutine en niet als hoofdprogramma. Komt de computer nu op deze manier bij regel 260 dan vindt hij daar een RETURN, zonder dat hij eerst een GOSUB die daar bijhoort is tegen gekomen. Dit leidt tot *foutmelding nummer 16.*
Zorg er altijd voor dat de subroutines achter het hoofdprogramma staan, en dat de computer niet per ongeluk na het doorlopen van het hoofdprogramma (met vertakkingen) aan de subroutines kan beginnen. Plaats een END opdracht.

Het is mogelijk om een bepaalde subroutine verschillende keren aan te roepen vanaf verschillende programmaregels. Dit is vooral handig als een bepaalde

taak verschillende keren gedaan moet worden. De computer houdt zelf bij vanaf welke regel hij naar de subroutine sprong en waar hij na de subroutine weer moet terugkeren.

```
5 REM WORTELTREKKEN
10 GOSUB 150:REM INSTRUCTIES
20 PRINT "UW GETAL S.V.P. (+ RETURN)";
30 INPUT GE
40 GOSUB 200:REM AFDRUKKEN
50 BEREK=GE/2
60 NB=(BEREK+(GE/BEREK))/2
70 GOSUB 200
80 IF (NB-BEREK)=0 THEN PRINT "BESTE SCHATTING:":PRINT :GOSUB 200:END
90 BEREK=NB
100 GOTO 60
110 END
150 PRINT :PRINT :PRINT "Dit progr. maakt gebruik van een"
155 PRINT "schattingsmethode die NEWTON"
160 PRINT "uitgevonden heeft."
165 PRINT "Het is een zeer snelle wijze om een"
170 PRINT "wortel van een getal te schatten."
180 PRINT "Zodra de comp. een getal heeft met een"
185 PRINT "naukeurigheid gelijk aan die van de "
190 PRINT "comp., wordt het getal afgedrukt.":PRINT :PRINT
195 RETURN
200 PRINT "getal: ";GE;"      SCHATTING: ";BEREK
210 PRINT
220 RETURN
```

U ziet hoe in dit programma een keer naar de subroutine op regel 150 en verschillende keren naar de subroutine die begint op regel 200 wordt gesprongen.
Het programma gebruikt een schattingsmethode van Newton om zo snel mogelijk de wortel van een door de gebruiker ingevoerd getal te berekenen. Zodra de nauwkeurigheid van de schatting groter is dan de nauwkeurigheid van de computer kan de computer geen verschil zien tussen twee verschillende schattingen. De nauwkeurigheid van beiden is immers groter dan de computer aan kan. De computer ziet de beide getallen als gelijk, zodat het verschil van de twee getallen gelijk is aan 0 (volgens de computer). Daarmee is de grootst mogelijke nauwkeurigheid bereikt. De computer springt uit de lus en geeft het resultaat weer.

De belangrijkste kenmerken van subroutines zijn:

- het zijn aparte programma-onderdelen
- ze zijn verschillende keren vanaf verschillende plaatsen aanroepbaar
- ze worden aangeroepen door GOSUB
- ze worden verlaten door RETURN
- variabelen uit het hoofdprogramma kunnen in de subroutine gebruikt worden

Kiezen tussen subroutines

Naast de hierboven genoemde manier om de GOSUB opdracht te gebruiken kent de Atari ook de opdracht:

ON ... GOSUB (bij .... ga naar subroutine...)

Deze opdracht laat de computer springen naar een subroutine die wordt aangegeven door een getal achter ON.
Stel u heeft een hele serie waarden en u wilt de computer aan de hand van deze waarden naar een bepaalde subroutine laten springen. Bijvoorbeeld:

```
10 GRAPHICS 0
20 PRINT :PRINT "GEHEEL GETAL ( >0 <8 )";
30 INPUT A
40 PRINT
50 ON A GOSUB 100,150,200,300,400,450,525
60 PRINT
70 GOTO 20
80 END
99 REM EERSTE SUBROUTINE
100 PRINT "DEZE SUBROUTINE BEGINT OP 100"
110 PRINT "U KOOS DUS GETAL 1"
120 RETURN
149 REM TWEEDE SUBROUTINE
150 PRINT "DIT IS REGEL 150"
160 PRINT "U KOOS GETAL 2"
170 RETURN
199 REM DERDE SUBROUTINE
200 PRINT "U HEEFT DRIE GEKOZEN"
210 RETURN
299 REM VIERDE SUBROUTINE
300 PRINT "U DRUKTE EEN 4 IN"
310 RETURN
399 REM VIJFDE SUBROUTINE
400 PRINT "HET PROGRAMMA IS AL BIJ REGEL 400."
410 PRINT "5 WAS DUS HET GETAL"
420 RETURN
449 REM ZESDE SUBROUTINE
450 PRINT "EEN ZES WAS UW KEUZE"
460 RETURN
524 REM ZEVENDE SUBROUTINE
525 PRINT "DE LAATSTE SUBROUTINE OP RGL. 525"
530 PRINT "U KOOS VOOR GELUKSGETAL 7"
540 RETURN
```

Het hoofdprogramma loopt van regel 10 tot en met regel 80.
In regel 30 wordt om een geheel getal tussen 1 en 7 gevraagd.
Regel 50 zorgt ervoor dat er afhankelijk van de ingetikte waarde naar een bepaalde subroutine wordt gesprongen. Naar aanleiding van de ingetikte waarde wordt gesprongen naar de subroutine die begint op het regelnummer dat als eerste, tweede, derde, vierde, vijfde, zesde of zevende achter de GOSUB-opdracht is genoemd. U kunt daar zelf de regelnummers voor kiezen. U moet er alleen wel op letten dat de regel waarnaar verwezen wordt ook daadwerkelijk bestaat.
Als u een waarde intikt van 0 of een waarde groter dan het aantal regelnummers achter de GOSUB opdracht, gaat de computer gewoon door met de volgende regel en wordt de gehele ON..GOSUB opdracht verwaarloosd.
Vult u een negatieve waarde of een getal groter dan 255 in (dat wil zeggen een getal dat niet meer in een byte geheugenruimte past), dan krijgt u *foutmelding 3* te zien.
Vult u een decimale breuk in, dan rondt de computer het getal af naar het dichtst bijzijnde getal en neemt deze afgeronde waarde als uw ingetikte waarde.

De waarde achter ON mag een variabele, een getal of een berekening zijn. Als het resultaat van de berekening of de waarde van de variabele geen geheel getal is, wordt de waarde afgerond. Voor waarden kleiner dan 0 of groter dan 255 geldt hetzelfde als hierboven.
Het terugkeren vanuit de subroutine gebeurt op dezelfde wijze als bij alle andere subroutines: met een RETURN opdracht. Door deze opdracht gaat de computer verder met de regel volgend op de regel met de ON..GOSUB.. opdracht.
Let in het voorbeeldprogramma op het gebruik van de REM opdrachten. Deze zijn allemaal onmiddellijk voor het begin van de subroutine geplaatst. Sommige programmeurs vinden dat alle REM's uit een programma gehaald moeten worden omdat ze onnodig het programma verlengen en geheugenruimte vergen. Door de REM's telkens vlak voor de subroutine te plaatsen, wordt het programma niet beinvloed.
Normaal gesproken is het echter beter om de REM's te laten staan, daar zij de opbouw van het programma toelichten.

ON...GOTO...

Naast de hierboven beschreven ON..GOSUB.. opdracht kent de Atari ook de ON..GOTO.. opdracht. De werking hiervan is gelijk aan die van ON..GOSUB.. met dat verschil dat de computer niet naar een subroutine springt vanwaar hij weer terugkeert, maar alleen naar een bepaalde programmaregel springt. Zonder nadere opdrachten in het programma springt de computer niet meer terug naar de regel volgend op de ON..GOTO.. opdracht.
Het verschil is gelijk aan het verschil tussen GOSUB en GOTO. GOSUB zorgt voor een sprong naar een ander deel van het programma waarna de computer weer terugspringt. GOTO zorgt slechts voor een sprong, zonder terugkeer.

Het volgende programma is een klein grafisch programma om driehoeken op verschillende manieren op het scherm te krijgen. Behalve de tekenopdrachten, zijn alle opdrachten eerder in het boek uitgelegd. De tekenopdrachten komen in de nu volgende hoofdstukken aan de orde.

```
5 DIM X(3),Y(3)
10 GRAPHICS 0
20 PRINT :PRINT "KLEUR (1) OF ZW/W (2)";:INPUT KL
30 PRINT :PRINT "BEGINPUNT X,Y";:INPUT X,Y
40 PRINT :PRINT "VERANDERING X RICHT.(1-10)";:INPUT DX
50 PRINT :PRINT "VERANDERING Y RICHT.(1-10)";:INPUT DY
60 ON KL GOTO 70,80
70 GRAPHICS 11:GOTO 100
80 GRAPHICS 8+16
100 REM
105 TRAP 1000
110 FOR LU=1 TO 20
120 DX=DX+DX/5:DY=DY+DY/5
130 GOSUB 500
140 NEXT LU
150 GOTO 140
160 END
500 REM TEKENEN
505 COLOR LU
510 X(1)=X+DX:Y(1)=Y-DY
520 PLOT X(1),Y(1)
530 X(2)=X-DX:Y(2)=Y+DY
540 DRAWTO X(2),Y(2)
550 X(3)=X+DX:Y(3)=Y+DY
560 DRAWTO X(3),Y(3)
570 DRAWTO X(1),Y(1)
580 RETURN
1000 FOR WA=1 TO 900:NEXT WA
1010 PRINT "DE TEKENING LOOPT VAN HET SCHERM"
1020 PRINT "PROBEER NOG EENS."
1030 FOR WA=1 TO 900:NEXT WA
1040 GOTO 10
```

14. DE SCHERMEN

Een van de beste kanten van de Atari computers is de mate van grafische mogelijkheden die alle Atari's met elkaar gemeen hebben. Hoewel er onderlinge verschillen aan te wijzen zijn tussen de verschillende types, hebben allen gemeen dat ze de gebruiker in staat stellen, indrukwekkende grafische afbeeldingen te maken.
De Atari kent verschillende grafische 'schermen'.. U moet dit beschouwen als combinaties van papier en pen. De ene schermmodus (=vorm waarin het scherm gepresenteerd wordt) is als een glad stuk papier met een zeer fijne viltstift; u kunt dunne, fijne lijnen tekenen in een enkele kleur. De andere schermmodus is veel eerder als een goed stuk papier met 16 dikke viltstiften. U kunt een tekening maken met verschillende kleuren, maar de lijnen zijn minder dun, waardoor de tekening wat groffer wordt. De nieuwe Atari's hebben ook een grafisch scherm dat u zich het beste voor kunt stellen als een vel heel ribbelig, sterk zuigend papier, met een stel dikke stiften. Tekenen gaat uitstekend, maar de lijnen worden een beetje rafelig.
Naast de grafische schermmodi kent de Atari de tekstschermen, die voorgesteld kunnen worden als een vel papier en een schrijfmachine: alleen tekst is mogelijk. Tussen deze mogelijkheden in, bestaat nog een schermmodus waarbij alleen met letters of met een beperkt aantal grafische symbolen gewerkt kan worden.
U merkt het al: keuze genoeg.

Als u grafisch met de Atari wilt gaan werken is het belangrijkste BASIC woord:

GRAPHICS (grafiek)

Deze opdracht vertelt aan de computer welke schermmodus gekozen moet worden, en bepaalt daardoor hoe het scherm er uit komt te zien. Achter de GRAPHICS opdracht komt een spatie en vervolgens een positief geheel getal of een variabele of een berekening met als resultaat een positief geheel getal. Zo is GRAPHICS 5 toegestaan, maar ook A=5, GRAPHICS A of GRAPHICS 2+3.

De schermmodi

Afhankelijk van het bouwjaar van uw Atari, kan met de GRAPHICS opdracht uit negen, twaalf of zelfs zestien verschillende schermmodi gekozen worden. Atari heeft in de loop der jaren de chip die voor het beeldscherm gedeelte zorgt (de CTIA of GTIA), gewisseld. Gelukkig heeft men er daarbij voor gezorgd dat de twee types volledig 'compatible' (uitwisselbaar) zijn, zodat programma's die gebruik maken van het ene type chip net zo goed werken met de andere chip en vice versa. De nieuwe chips bieden alleen een paar schermmodi extra.

Een deel van de schermmodi kent ook nog eens twee verschillende verschijningsvormen: met en zonder 'tekstvenster'. Een tekstvenster is een gedeelte onderin het beeldscherm, waar de grafische afbeelding niet kan komen en die gereserveerd is voor teksten. Bijvoorbeeld foutmeldingen, INPUT opdrachten etc. Zo'n tekstvenster is naar keuze uit te schakelen.

De mogelijke schermmodi zijn: (niet elke Atari kent deze modi allemaal)

graphics	gebruik	resolutie met tekstvenster	resolutie zonder tekstvenster	kleuren	geheugengebruik met tekstvenster	geheugengebruik zonder tekstvenster
0	tekst	-	40x 24	1	-	992
1	tekst	20x 20	20x 24	5	674	672
2	tekst	20x 10	20x 12	5	424	420
3	grafiek	40x 20	40x 24	4	434	432
4	grafiek	80x 40	80x 48	2	694	696
5	grafiek	80x 40	80x 48	4	1174	1176
6	grafiek	·160x 80	160x 96	2	2174	2184
7	grafiek	160x 80	160x 96	4	4190	4200
8	grafiek	320x160	320x192	1	8112	8138
9	grafiek	-	80x192	1	-	8138
10	grafiek	-	80x192	9	-	8138
11	grafiek	-	80x192	16	-	8138
12	gra./tkst	40x 20	40x 24	5	1154	1152
13	gra./tkst	40x 10	40x 12	5	664	660
14	grafiek	160x160	160x192	2	4270	4296
15	grafiek	160x160	160x192	4	8112	8138

Uit deze tabel zijn verschillende belangrijke zaken af te lezen. Het is door de tabel een stuk duidelijker waarom er zoveel verschillende schermmodi zijn: elke schermmodus heeft zijn eigen mogelijkheden. Zo leidt het gebruik van fijnere lijnen meestal tot minder kleur. Daarnaast is in de laatste twee kolommen af te lezen dat hoe meer kleur en hoe fijnere lijnen u wenst, hoe meer geheugenruimte het maken van de schermodus kost. Dit kan oplopen tot 8138 geheugenplaatsen (dat wil zeggen bijna 8 Kbyte!).

Ruitjespapier

In de derde en vierde kolom ziet u de resolutie of het oplossend vermogen vermeld. Wat wil dat zeggen?
Het oplossend vermogen geeft aan hoe dun de lijnen op het scherm komen te staan en hoeveel lijnen er op het scherm passen.
U moet het maken van grafiek op een computer voorstellen als het maken van

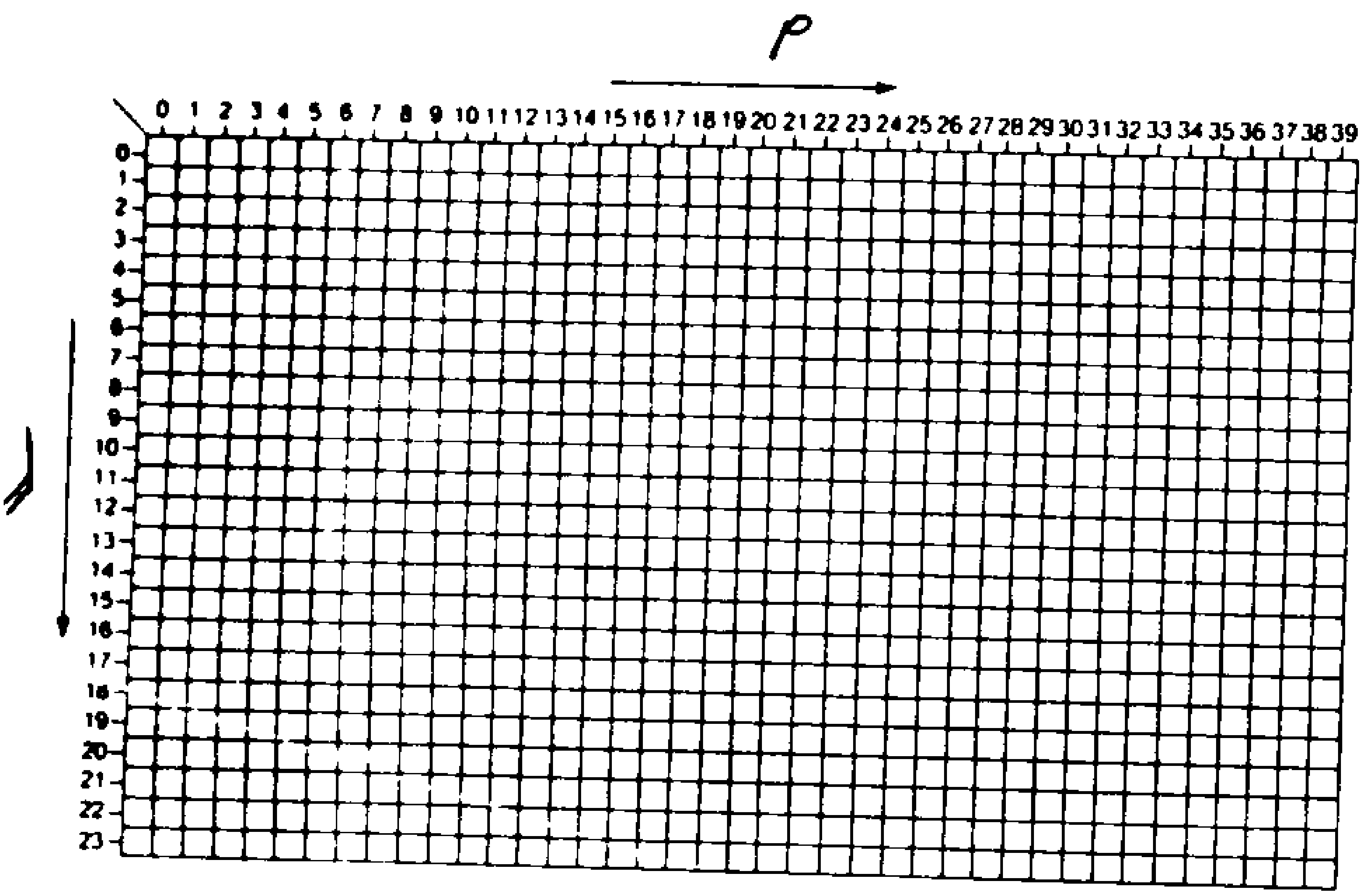
tekeningen op ruitjespapier. U mag alleen tekenen door verschillende hokjes al dan niet in te kleuren. Door een hele rij hokjes naast elkaar in te kleuren, maakt u een lijn. Een verzameling lijnen (dus een verzameling ingekleurde hokjes) vormen een tekening. Hoe kleiner de hokjes die ingekleurd moeten worden, hoe preciezer de tekening kan worden en hoe dunner de lijnen zijn. Vergelijkt u dit ook maar met foto's die u van de fotograaf terug krijgt en foto's in de krant. De foto's die u van de fotograaf terug krijgt bestaan uit heel veel kleine gekleurde stippeltjes (kijkt u er maar eens naar met een vergrootglas). Om druktechnische redenen is het echter bij een krant niet mogelijk om die kleine stippeltjes af te drukken. De foto in een krant bestaat dan ook uit veel grotere stippen. Toch geven beide soorten foto's door een verzameling al dan niet gekleurde stippen een grafisch beeld van de werkelijkheid.
Een computer kan ook alleen stippen weërgeven. Het aantal stippen op een bepaalde oppervlakte (dat wil zeggen, de maat van de stippen) bepaalt het oplossend vermogen.
Bij een computer wordt het aantal stippen van links naar rechts en het aantal stippen van boven naar onder (samen een rasterwerk vormend) aangegeven met twee getallen. Deze vindt u telkens in de derde en vierde kolom hierboven vermeld.

Het opdelen van het scherm alsof het een vel ruitjespapier is, heeft nog een voordeel. Door alle kolommen en alle rijen met ruitjes te nummeren, kunnen we elk punt op het raster door middel van twee getallen aanduiden.

Bij de Atari staat het eerste getal (de X-coordinaat) voor het kolomnummer en het tweede getal (de Y-coordinaat) voor het nummer van de rij. In de tekst van dit boek zullen we een bepaald punt op het ruitjespapier, dat door de schermmodus gevormd wordt, telkens aanduiden met twee getallen tussen haakjes. Zo is (2,3) het hokje dat op de kruising van de tweede kolom en de derde rij ligt. (0,0) is het punt dat ligt op de kruising tussen de nulde kolom en de nulde rij. Dat is de linkerbovenhoek want zowel de kolommen als de rijen beginnen bij 0 te nummeren. (156,97) is het punt in de 156-ste kolom en 97-ste rij.

Tekstscherm 0

Schermmodus 0 is het meest gebruikte scherm. Het is tevens de zogenaamde **'default'** van de computer, dat wil zeggen dat bij het aanzetten van de computer of bij het ontdekken van een fout, de computer automatisch deze schermmodus in werking stelt. Als u bij een van de andere schermmodi een tekstvenster gebruikt, gedraagt dit tekstvenster zich als een klein stukje scherm 0. Scherm 0 heeft 40 kolommen en 24 rijen.



In scherm 0 is het altijd mogelijk teksten af te beelden met de gewone PRINT-opdracht (dit in tegenstelling tot alle andere schermmodi). De te PRINTen tekst verschijnt bij de tekstcursor en wordt aangegeven door een wit blokje. De plaats van de tekstcursor wordt bepaald door vorige PRINT opdrachten, leestekens en de POSITION-opdracht. *Sla hiervoor het hoofdstuk 'Invoer en beeldweergave' na.*
Scherm 0 kent standaard een linker kantlijn die twee posities breed is. Als u de POSITION opdracht gebruikt, wordt er niet gelet op de kantlijn. Zowel de linker als de rechterkantlijn van scherm 0 zijn door de gebruiker in te stellen. Daarvoor moet door middel van een POKE-opdracht een waarde direkt in een geheugenadres geplaatst worden.

POKE 82, kolomnummer van linkerkantlijn
POKE 83, kolomnummer van rechterkantlijn

In adres 82 kunt u een waarde van 0 tot en met 39 POKEn, in adres 83 idem. De waarde van adres 83 moet echter groter zijn dan die van adres 82. Het volgende programma laat zien hoe u door middel van deze POKE opdrachten het scherm steeds smaller kunt laten worden. Let u ook op de wijze waarop de READY melding na het programma getoond wordt.

```
10 FOR A=0 TO 20
20 POKE 82,A:POKE 83,40-A
30 PRINT
40 FOR B=1 TO 100
50 PRINT B;
```

```
60 NEXT B
70 PRINT
80 NEXT A
```

Om het er allemaal nog vreemder uit te laten zien kunt u nog een POKE opdracht gebruiken. Voeg toe:

```
15 POKE 755,4
```

Door in adres 755 de waarde 4 te POKEn, worden alle lettertekens op hun kop geplaatst. Een vreemde eigenschap die stamt uit de tijd dat Atari vooral spelmachines maakten die het beeld via een spiegel projecteerden (voor gebruik in grotere spelmachines in cafe's enzovoort).

Naast de gebruikelijke PRINT-opdrachten kunt u ook allerlei grafische opdrachten gebruiken in tekstmodus 0. Omdat het echter om een tekstmodus gaat, worden er **alleen** lettertekens geplaatst. De volgende grafische instrukties zullen nog ruim behandeld worden in de volgende hoofdstukken. Ze worden hier kort aangehaald om de mogelijkheden van scherm 0 te laten zien.

De PLOT (=teken) opdracht wordt bij de grafische schermen gebruikt om een punt op een bepaalde plaats te zetten. De plaats wordt aangegeven door twee getallen achter de PLOT opdracht. Het eerste getal geeft de kolom aan (X-coordinaat), het tweede getal geeft de rij aan (Y-coordinaat).
In modus 0 werkt de PLOT opdracht vergelijkbaar. Nu wordt er echter geen punt geplaatst op de aangegeven positie, maar een letterteken. Welk letterteken wordt afgedrukt is afhankelijk van de waarde die meegegeven is aan de COLOR (=kleur) opdracht. Deze opdracht wordt in de grafische modi gebruikt om de kleur van de te tekenen lijn op te geven. In modus 0 wordt dezelfde opdracht gebruikt om het af te drukken letterteken op te geven.

```
10 GRAPHICS 0
20 COLOR 70
30 PLOT 10,10
```

Dit programma drukt op positie (10,10) een letterteken af met de ASCII code 70 (een F).
Nog opvallender is de mogelijkheid om de DRAWTO (=teken naar) opdracht te gebruiken in modus 0. Deze opdracht wordt in de grafische modi gebruikt om een lijn te trekken naar een door de twee coordinaten opgegeven punt. Ook nu weer bepaalt de COLOR-opdracht in welke kleur de lijn getrokken zal worden. In scherm 0 werkt de opdracht vergelijkbaar. In plaats van in een bepaalde kleur getrokken te worden, wordt de lijn samengesteld uit een verzameling van lettertekens die wordt bepaald door de ASCII-code achter de COLOR opdracht.

```
10 GRAPHICS 0
20 COLOR 70
30 PLOT 5,5
40 DRAWTO 21,12
```

Probeer van beide bovenstaande programma's zowel de waarde achter COLOR als de waarden achter PLOT en DRAWTO te veranderen. Kijk welke veranderingen u op het scherm ziet.

Grafische symbolen

Ook in modus 0 is het mogelijk tekeningen te maken met behulp van allerlei grafische symbolen. Deze symbolen vindt u bij de Atari 130XE aan de voorkant van de toetsen.

De grafische symbolen kunt u direkt op het scherm krijgen door de gewenste toets *samen* met de CONTROL-toets in te drukken. De grafische symbolen kunnen in een programma opgenomen worden door hen binnen een string te plaatsen. Daarvoor drukt u tijdens het intikken van een string de CONTROL-toets samen met de gewenste toets in:

A$="CONTROL-S"

Dit dient niet gelezen te worden als A$ is gelijk aan C O N T R O L - S, maar als A$ is gelijk aan de CONTROL-toets en de S toets tegelijk ingedrukt. Dat wil zeggen: een + teken. A$ kan nu verder in het programma gebruikt worden. Op het scherm zal het er telkens uitzien als een +.

De grafische symbolen kunnen ook afgedrukt worden door middel van een CHR$ opdracht. Het volgende programma laat de verschillende grafische tekens van modus 0 zien.

```
10 GRAPHICS 0
20 FOR A=0 TO 31
30 PRINT CHR$(A);" ";
40 NEXT A
50 FOR A=126 TO 159
60 PRINT CHR$(A);" ";
70 NEXT A
```

De codes die nodig zijn voor het afdrukken van de grafische tekens vallen niet onder de internationale ASCII-set. De ASCII-set van de Atari wordt dan ook wel ATASCII-set genoemd.

In het programma wordt code 123 weggelaten, daar deze voor het wissen van het scherm zorgt, zodat een deel van de tekens niet meer zichtbaar is. Met enkele kleine veranderingen kunt u vast spieken bij de mogelijkheden van enkele andere schermmodi. Verander zowel regel 30 als 60 in:

PRINT #6;CHR$(A);" "

Door het programma enkele malen te laten runnen met in regel 10 achtereenvolgens GRAPHICS 0, GRAPHICS 1, GRAPHICS 2, GRAPHICS 12 en GRAPHICS 13, kunt u de verschillen tussen de modi goed zien. De modi 12 en 13 zorgen voor een wat rafelig beeld. De modi 1 en 2 lijken wel echte tekstmodi te zijn: zonder speciale opdracht worden de grafische symbolen als normale letters weergegeven. Zie echter de paragraaf over de modi 1 en 2.

De kleur van scherm 0

Bij het aanzetten van de computer heeft scherm 0 altijd een lichtblauwe achtergrond, witte letters en een zwarte rand. Dit zijn de standaardkleuren omdat in het algemeen de combinatie lichtblauw/wit op kleurentelevisies het scherpste beeld geeft. Het is heel goed mogelijk deze kleuren te veranderen. Ook hiervoor gebruikt u een opdracht die normaal gesproken vooral gebruikt wordt in de grafische schermen:

SETCOLOR reg,kleur,sterkte

De SETCOLOR (=kleurinstelling) opdracht kan verschillende kleurregisters instellen. De Atari heeft vijf verschillende registers waar informatie over kleur en helderheid is opgeslagen. De informatie in de verschillende registers wordt echter door elke schermmodus anders gebruikt. In modus 1 beinvloeden de waarden in de registers de volgende zaken:

register 0	niet gebruikt
register 1	helderheid van lettertekens
register 2	kleur van lettertekens en achtergrond
register 3	niet gebruikt
register 4	kleur van rand

De Atari maakt onderscheid tussen twee aspecten van kleur: de werkelijke kleur en de helderheid.
Zoals in bovenstaande tabel te zien is, is het in modus 0 onmogelijk om de letters een andere kleur te geven als de achtergrond (register 2 regelt gelijktijdig de kleur van de lettertekens en van de achtergrond). Het verschil in de voor de ogen waarneembare kleur ligt dus helemaal in het helderheidsverschil. Dit kan door register 1 worden bepaald. Kijk met het volgende programma hoe de helderheid van de lettertekens veranderd kan worden.

```
10 GRAPHICS 0
20 FOR A=0 TO 15
30 PRINT :PRINT "HELDERHEIDSTEST"
40 SETCOLOR 1,7,A
50 FOR WA=1 TO 900:NEXT WA
60 NEXT A
```

Valt het u op dat op een gegeven moment alle letters verdwijnen? De helderheid van de achtergrond is gelijk aan de achtergrond van de lettertekens. Merk ook op dat slechts een op de twee keer dat de tekst afgedrukt wordt, er een verandering in helderheid te zien is. Hoewel de waarden voor helderheid van 0 tot en met 15 lopen, zijn alleen **0** en de **even** waarden van belang.

Met een kleine verandering kan ook de kleur van het beeld worden veranderd. Verander regel 40:

```
40 SETCOLOR 2,A,2
```

Door deze regel nogmaals te veranderen kunt u ook de kleur van de rand laten veranderen:

```
40 SETCOLOR 2,A,2 : SETCOLOR 4,15-A,8
```

In plaats van de SETCOLOR opdracht kunt u ook een waarde in bepaalde geheugenadressen POKEn. Daarbij moet een kleine rekensom gemaakt worden. In het geheugenadres kan immers maar een waarde geplaatst worden, terwijl de SETCOLOR opdracht twee waarden (behalve het nummer van het register) verwerkt. De registers zijn op de volgende adressen te vinden:

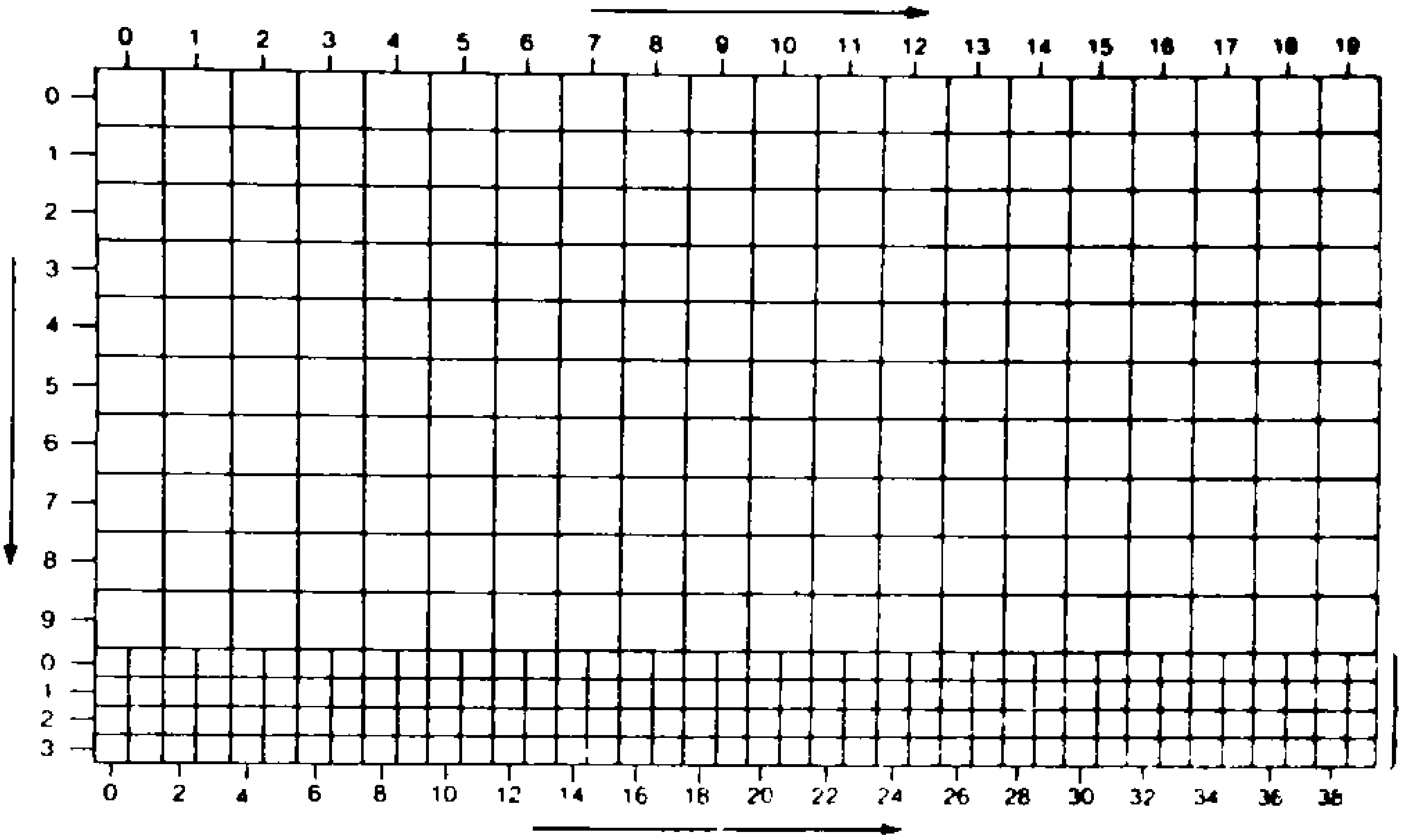
register 0	POKE 708,
register 1	POKE 709,
register 2	POKE 710,
register 3	POKE 711,
register 4	POKE 712,

De waarde die in het adres gePOKEd moet worden vindt u door de kleurwaarde met 16 te vermenigvuldigen en de helderheidswaarde daarbij op te tellen:

te POKEn waarde = 16 * kleur + helderheid

Tekstschermen 1, 2, 12, en 13

De schermmodi 1 en 2 worden 'opgeblazen' tekstschermen genoemd. De lettertekens worden in modus 1 twee maal zo breed als in modus 0 weergegeven. In modus 2 worden de lettertekens twee maal zo hoog en twee maal zo breed weergegeven. De grootte van het raster (de resolutie) is dan ook afhankelijk van de gebruikte modus: In de breedte gaan evenveel kolommen, maar in de hoogte heeft modus 2 slechts half zoveel rijen als modus 1.



Onder het eigenlijke scherm van modus 1 en 2 bevindt zich een tekstvenster van 40 posities breed en 4 regels hoog. Dit tekstvenster gedraagt zich als scherm 0. Zie daar voor nadere informatie. Om tekst op het eigenlijke scherm te krijgen moet de opdracht

PRINT #6

gebruikt worden. In het hoofdstuk 'Bestanden verwerken' heeft u kunnen lezen dat er verschillende IOCB kanalen bestaan. Kanaal 6 is het kanaal dat gebruikt wordt voor grafische opdrachten. Het kanaal wordt automatisch geopend en gesloten.
Dit automatisme is zo sterk dat bij typisch grafische opdrachten als PLOT en DRAWTO, de kanaalaanduiding helemaal overbodig is. De computer gaat er bij deze opdrachten automatisch vanuit dat ze voor het scherm bedoeld zijn. De PRINT-opdracht kan echter zowel op het tekstvenster als op het scherm slaan. Gebruikt u PRINT zonder toevoeging, dan wordt de tekst in het tekstvenster geplaatst.

Gebruikt u PRINT #6, dan wordt de tekst in het scherm geplaatst.
Net als in scherm 0 kunt u door middel van POSITION een plaats bepalen op het scherm waar de tekst geplaatst moet worden. PRINTen kan gebeuren door:

```
PRINT #6; A$
PRINT #6;"string"
PRINT #6; CHR$(70)
```

```
PRINT #6; VARIABELE
PRINT #6; BE+REK+ENING
```

Het volgende programma laat een deel van de mogelijke lettertekens in modus 1 en 2 zien.

```
10 GRAPHICS 1
20 FOR Z=1 TO 2
30 FOR A=1 TO 122
40 PRINT #6;CHR$(A);
50 NEXT A
60 FOR WA=1 TO 5000:NEXT WA
70 GRAPHICS 2
80 NEXT Z
```

Ziet u dat er slechts een beperkt aantal tekens mogelijk is? De computer herhaalt de tekens, maar in een andere kleur. Alle tekens zijn opgeslagen in een beperkt aantal codes. Deze codes bepalen niet alleen welke tekens u krijgt, maar ook in welke kleur. Door in bovenstaand programma in te vullen:

```
30 FOR A=32 TO 95
```

krijgt u de helft van de tekens in rood te zien.
De codes 160 tot en met 223 zorgen voor de helft van de tekens in blauw. De codes 0 tot en met 31 en 96 tot en met 127 zorgen voor de helft van de tekens in geel (deze serie codes is gesplitst!).
De codes 128 tot en met 159 en 224 tot en met 255 geven paarse tekens (gesplitste serie codes!).
Hoe komen we aan de andere helft van de mogelijke tekens? Daarvoor moet weer direkt in het geheugen worden ingegrepen.

```
POKE 756,226
```

Door in geheugenplaats 756 de waarde 226 te POKEn, kiest u de alternatieve lettertekenset van de computer. Dit geeft andere tekens in schermmodi 1 en 2. U keert weer terug naar normale tekens met

```
POKE 756,224
```

Voeg aan bovenstaand programma toe:

```
5 POKE 756,226
```

En run het programma nogmaals.

Een bijzonder effect krijgt u door eerst het scherm op te roepen GRAPHICS 1 of GRAPHICS 2 en daarna pas de karakterset te veranderen. Probeer dit uit in het programma door regel 5 te verwijderen en opnieuw te plaatsen. maar dan als regel 15. Ziet u het verschil? De computer vult alle spaties die er al waren met hartjes. Hoe romantisch kan een computer toch zijn!

Verschil modus 0 en modi 1 en 2

Naast de hierboven genoemde verschillen tussen het 'default' scherm 0 en de tekstschermen 1 en 2, zijn er nog enkele noemenswaardige verschillen:

- De modi 1 en 2 kennen geen standaard ingestelde kantlijn.
- Zowel modus 0, als het tekstvenster kennen het 'scroll' effect. Dat wil zeggen dat als er geen tekst meer op het scherm past. de computer de bovenste regel van het scherm haalt, alle regels een rij naar boven verplaatst en onderaan het scherm een nieuwe regel plaatst. In het tekstvenster gaat dit zelfs erg snel. omdat er maar vier regels zijn. De modi 1 en 2 kennen dit effekt niet. Als u meer regels probeert te PRINTen dan er op het scherm aanwezig zijn. zal de computer het programma stoppen met foutmelding 141. Deze foutmelding wil zeggen dat u de computer opdracht heeft gegeven iets af te drukken op een plaats buiten het scherm.
- Modus 0 kan de gehele internationale ASCII-set aan: modi 1 en 2 slechts de halve ASCII-set. U kunt daar gebruik van maken bij het weergeven van teksten op deze schermen. Bij het intikken van een string die door PRINT #6 afgedrukt gaat worden, kunt u ervan uit gaan dat, zonder speciale opdrachten, kleine letters als gele hoofdletters weergegeven zullen worden en hoofdletters als rode hoofdletters weergegeven zullen worden. Zie ook de vorige alinea's.
- POSITION bepaalt wel de plaats van de tekst in het hoofdscherm. maar niet van de tekst in het tekstvenster. Het is wel mogelijk de plaats te bepalen in het tekstvenster, maar daarvoor is de POKE opdracht weer nodig.

POKE 656, rij
POKE 657, kolom

Met deze twee adressen kunt u de kolom en rij opgeven waar de tekstcursor van het tekstvenster geplaatst wordt. Daar kunt u vervolgens de door u gewenste tekst PRINTen. Let op! De bovenste rij van het tekstvenster is weer 0, ondanks het feit dat dit een tamelijk lage positie op het scherm is. De onderste regel van het tekstvenster is nummer 3.

Kleuren in modi 1 en 2

Ook in de schermen 1 en 2 zijn de kleurregisters van belang voor hetgeen er op het scherm komt te staan.

register 0	hoofdletters, cijfers en leestekens
register 1	kleine letters en grafische symbolen
register 2	inverse van tekens van register 0
register 3	inverse van tekens van register 1
register 4	achtergrondkleur

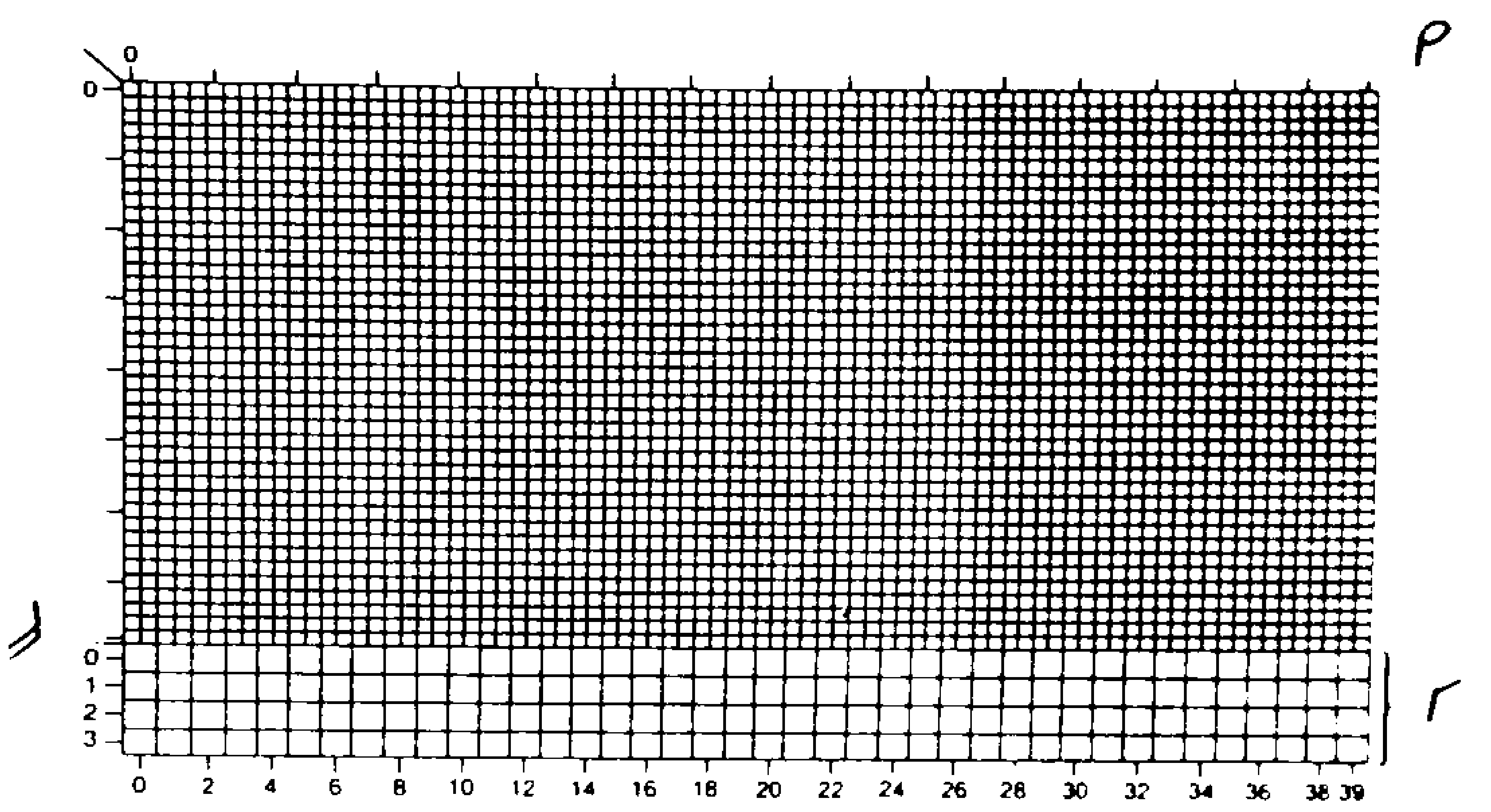
De modi 1 en 2 kunnen alle vijf registers gebruiken. Er zijn echter maar vijf registers, en het tekstvenster gebruikt er ook drie. Het tekstvenster gedraagt zich immers precies als scherm 0. Het is niet mogelijk de waarde van bijvoorbeeld register 1, een andere waarde voor scherm 1 en 2 (kleine letters en grafische symbolen) te laten hebben dan voor het tekstvenster (helderheid van de letters). Bepaalde instellingen die gewenst zijn voor effekten op het hoofdscherm, beinvloeden automatisch de helderheid van de lettertekens in het tekstvenster.

Schermen 12 en 13

Deze tekstschermen hebben twee keer zoveel kolommen als de tekstschermen 1 en 2. Omdat de tekens echter vergroot worden weergegeven, lopen deze in elkaar over. Daarbij komt dat de randen van de tekens een beetje raffelig worden en de kleuren gemengd zijn. Deze schermen zijn vooral bedoeld voor speciale effecten als aankondigingsbladzijden van een programma.
Overigens werken deze schermen als hierboven aangegeven.

De vier-kleuren schermen 3, 5, 7 en 15

De schermen 3, 5, 7 en 15 hebben gemeen dat het grafische schermen zijn met de mogelijkheid tot het weergeven van vier kleuren. Onder elk scherm bevindt zich het tekstvenster. Dit gedraagt zich weer als scherm 0.
In tegenstelling tot de tekstvensters zijn de grafische vensters bedoeld voor het weergeven van tekeningen. Het weergeven van teksten is dan ook niet goed mogelijk. Daar staat tegenover dat de Atari u de mogelijkheid biedt om tekeningen met fijne lijnen te trekken. Met scherm 3 kunt u tamelijk dikke lijnen trekken, met scherm 5 en 7 iets dunner en met scherm 15 de dunste lijnen. Scherm 15 kost echter wel het meeste geheugenruimte. Maar liefst 8112 bytes (als u een tekstvenster gebruikt). Dat wil zeggen dat bij een deel van de Atari's dit scherm niet voorkomt. Hetzelfde geldt echter voor scherm 7 dat nog altijd 4190 bytes vraagt. Dit scherm is wel altijd bruikbaar, maar als u een erg lang programma heeft, loopt u de kans dat het niet meer in het geheugen past, doordat het scherm teveel geheugen vergt. U zult zelf altijd bij het programmeren de keuze moeten maken tussen een korter programma, of een eenvoudiger schermweergave (of indien bij uw type Atari mogelijk: het uitbreiden van de geheugencapaciteit).



In de tekening ziet u weergegeven hoe groot het verschil in fijnheid (oplossend vermogen of resolutie) van de verschillende schermen is:

scherm 3:	40 kolommen bij 20 rijen(+ tekstvenster)
scherm 5:	80 kolommen bij 40 rijen(+ tekstvenster)
scherm 7:	160 kolommen bij 80 rijen(+ tekstvenster)
scherm 15:	160 kolommen bij 160 rijen(+ tekstvenster)

U moet de afmetingen van het scherm, uitgedrukt in kolommen en rijen, bij het programmeren altijd goed in de gaten houden. Als u probeert iets buiten het scherm af te drukken, volgt een foutmelding. Zo zal de volgende PLOT (=teken) opdracht wel werken in scherm 7, maar niet in scherm 3:

```
GRAPHICS 7 : PLOT 150,30      goed
GRAPHICS 3 : PLOT 150,30      foutmelding 141
```

Grafische opdrachten

Waar u bij de tekstvensters gebruik maakte van tekstopdrachten (als PRINT) om iets op het scherm te zetten, moet u op de grafische schermen tekenopdrachten gebruiken om iets weer te geven.
Om een alledaags voorbeeld te geven: het maken van teksten doet u door middel van een schrijfmachine en het maken van tekeningen doet u door middel van potlood en bijvoorbeeld een penseel. Het is in beperkte mate mogelijk om met behulp van een penseel, tekeningen als illustratie tussen de

teksten te plaatsen (zie daarvoor het volgende hoofdstuk), maar het is zeer lastig om met de schrijfmachine teksten te plaatsen op het linnen van het schilderij. De grafische opdrachten die de Atari u biedt zijn:

POSITION kolom, rij	**(= positie)**
PLOT kolom, rij	**(= teken)**
DRAWTO kolom, rij	**(= trek lijn naar)**
LOCATE kolom, rij, x	**(= localiseer)**
COLOR x	**(= kleur)**
SETCOLOR reg, kleur, helder	**(= kleurinstelling)**

De woorden kolom en rij slaan telkens op de twee coordinaten die bij de opdracht opgegeven moeten worden om de computer te laten weten waar een punt gezet moet worden, of waar een lijn naar toe getrokken moet worden. De enige nieuwe opdracht is de LOCATE (=localiseer) opdracht. Deze opdracht bepaalt de kleur van het opgegeven punt. Hiervoor kijkt de computer voor u in dat deel van het geheugen dat gereserveerd is voor de schermopmaak. In de schermmodi met tekst, bepaalt de LOCATE opdracht welk letterteken op de opgegeven positie staat.
De verschillende grafische opdrachten en het gebruik ervan zullen uitgebreid in de komende hoofdstukken aan de orde komen.

Kleuren in de vier-kleuren modi

De vier-kleuren modi delen drie van de vijf kleurregisters met scherm 0 (het tekstvenster). Het is niet mogelijk om bepaalde combinaties van tekst in het tekstvenster en kleuren in de tekening te maken.

De vier-kleuren modi gebruiken de kleurregisters als volgt:

register 0	COLOR 1
register 1	COLOR 2
register 2	COLOR 3
register 3	niet gebruikt
register 4	COLOR 0, achtergrondkleur

Met deze tabel is een interessant gegeven van de werking van de COLOR opdracht gegeven. Achter COLOR staat in de tabel telkens een nummer, geen kleur. COLOR 1 weergeeft de kleur die in register 0 staat en niet een bepaalde vastliggende kleur. Met andere woorden: ***er kunnen weliswaar slechts vier kleuren tegelijk op het scherm geplaatst worden, maar die vier kleuren kunnen door de gebruiker zelf gekozen worden uit de zestien kleuren van de Atari.*** Dit zijn:

Kleurcode	Kleur
0	grijs
1	licht-oranje (goud)
2	oranje
3	rood-oranje
4	roze
5	paars
6	paars-blauw
7	blauw
8	lichter blauw
9	licht blauw
10	blauw-groen (turquoise)
11	groen-blauw
12	groen
13	geel-groen
14	oranje-groen
15	licht-oranje

U ziet, de Atari heeft een ware regenboog aan kleuren. Omdat kleur een subjectieve gewaarwording is, en elke kleurentelevisie anders is afgesteld, kunt u persoonlijk de kleuren anders ervaren als hierboven aangegeven.
Om de kleuren op uw eigen televisie te zien, kunt u het volgende programma runnen. Het maakt gebruik van een van de grafische modi, die in een van de volgende paragrafen wordt behandeld.

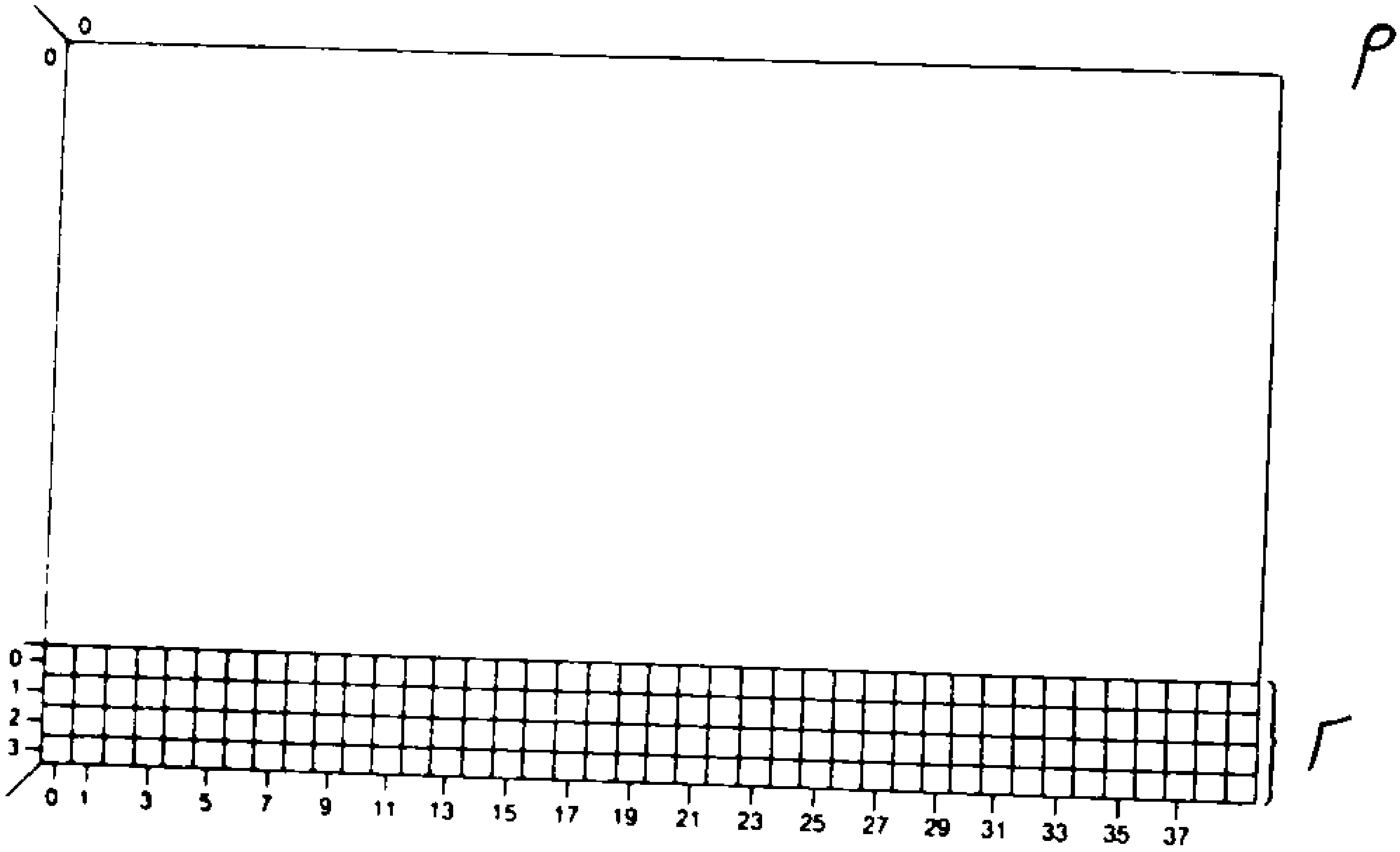
```
10 GRAPHICS 11
20 PLOT 40,176
30 FOR Z=90 TO 40 STEP -1
40 COLOR Z
50 FOR C=-1.5 TO 1.55 STEP 0.2
60 Y=190-2*Z*COS(C):X=40+(Z/2.5)*SIN(C)
70 DRAWTO X,Y
80 NEXT C
90 NEXT Z
100 GOTO 100
```

Twee kleuren-schermen 4, 6 en 14

Deze twee schermmodi hebben hun kleurinformatie anders opgeslagen dan de vier-kleuren modi. Er zijn slechts twee kleuren tegelijk op het scherm mogelijk, maar het oplossend vermogen is hoger:

scherm 4:	80 kolommen bij 40 rijen	(+ tekstvenster)
scherm 6:	160 kolommen bij 80 rijen	(+ tekstvenster)
scherm 14:	160 kolommen bij 160 rijen	(+ tekstvenster)

Ook bij deze schermen geldt, dat hoe fijner het oplossend vermogen is, hoe meer geheugenruimte de beeldopbouw vraagt. Scherm 14 vraagt 4270 bytes geheugenruimte (als het tekstvenster gebruikt wordt).



Kleuren

Bij deze schermen heeft u geen last van het samen gebruiken van registers door het tekstvenster en het grafische scherm. Slechts de achtergrondkleur (COLOR 0 of register 4) wordt door allebei gebruikt.

register 0	COLOR 1
register 1	niet gebruikt
register 2	niet gebruikt
register 3	niet gebruikt
register 4	COLOR 0 en achtergrond

U kunt dus COLOR 1 (register 0) gebruiken voor de tekeningen, zonder daarbij op de kleur van de tekst in het tekstvenster te hoeven letten.

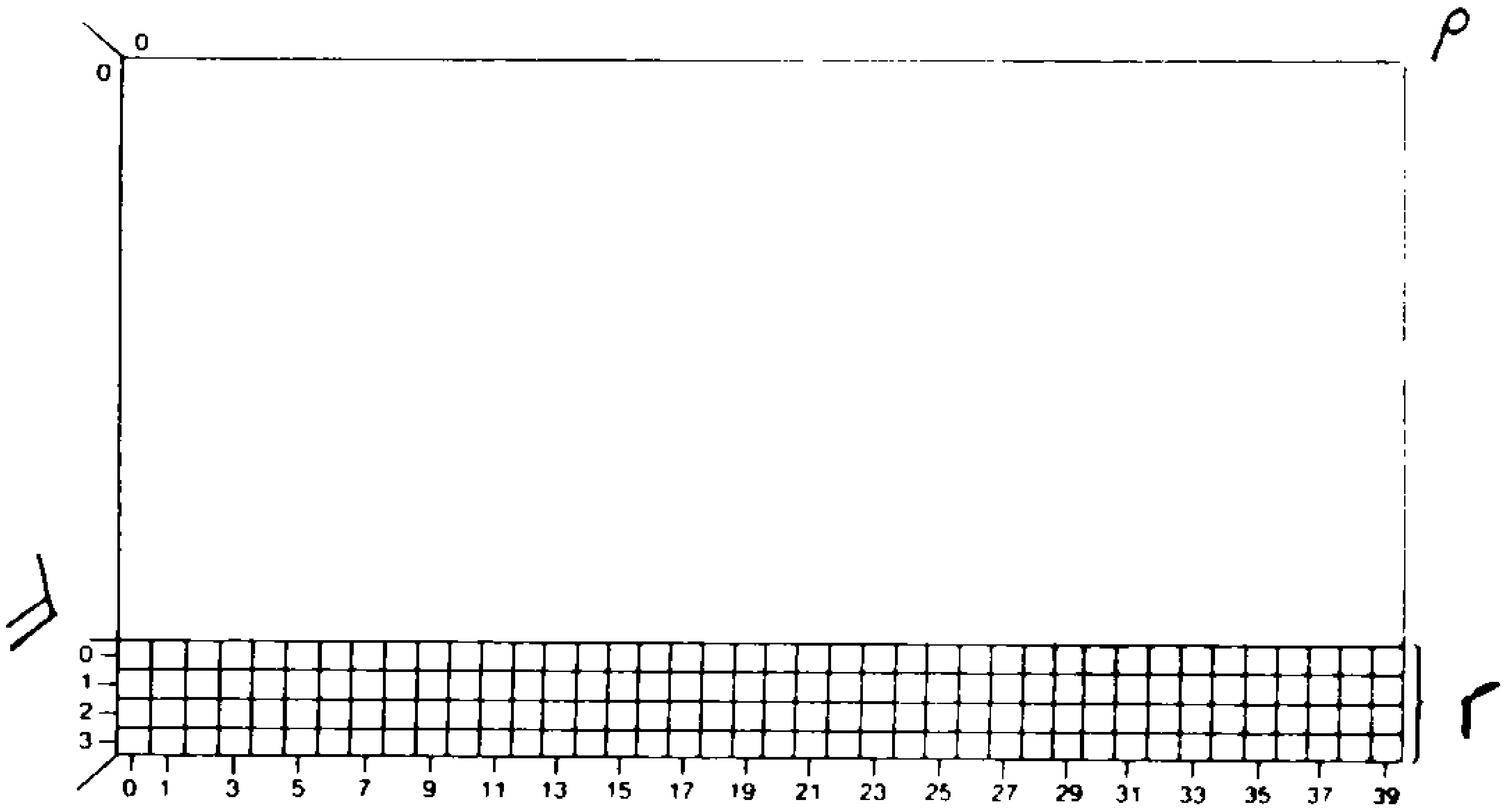
De kleur stelt u in door middel van:

SETCOLOR 0, kleur, helderheid
of
POKE 708, 16*helderheid + kleur

Twee-kleuren scherm 8

Dit scherm verschilt weinig van de andere twee-kleuren schermen. De opbouw van de kleuren is iets anders, en deze schermmodus heeft een fantastisch fijn oplossend vermogen:

scherm 8: 320 kolommen bij 160 rijen(+ tekstvenster)



De gebruik van de kleurregisters:

register 0	niet gebruikt
register 1	COLOR 1
register 2	achtergrond
register 3	niet gebruikt
register 4	kleur van de rand

Geen tekstvenster en wissen van het scherm

Alle tot nu toe genoemde tekst en grafische schermen hebben de beschikking over een tekstvenster. Het is mogelijk dit tekstvenster weg te laten en het scherm over het hele beeldscherm te laten lopen. Het enige nadeel is daarbij wel, dat als de computer een fout wil melden, het gehele scherm gewist wordt. Om dit te voorkomen moet u gebruik maken van de TRAP opdracht. Hetzelfde geldt voor het beeindigen van het programma. Aan het einde van het programma wil de computer de mededeling READY kwijt. Dit zorgt ervoor dat uw scherm wordt gewist. Dit is alleen te voorkomen door te zorgen dat het programma niet afloopt. Bijvoorbeeld door een oneindige lus:

```
100 GOTO 100
```

Om het tekstvenster te verwijderen, telt u bij de code van het scherm 16 op:

GRAPHICS 1+16	is gelijk aan	GRAPHICS 1 zonder tekstvenster
GRAPHICS 2+16	is gelijk aan	GRAPHICS 2 zonder tekstvenster
GRAPHICS 3+16	is gelijk aan	GRAPHICS 3 zonder tekstvenster
etc.		

Op dezelfde wijze kan het optellen van het getal 32, ervoor zorgen dat het beeld niet gewist wordt bij het wisselen van scherm:

GRAPHICS 1+32	is gelijk aan	GRAPHICS 1 zonder wissen
GRAPHICS 2+32	is gelijk aan	GRAPHICS 2 zonder wissen
etc.		

Combinatie van deze twee extra mogelijkheden kan ook:

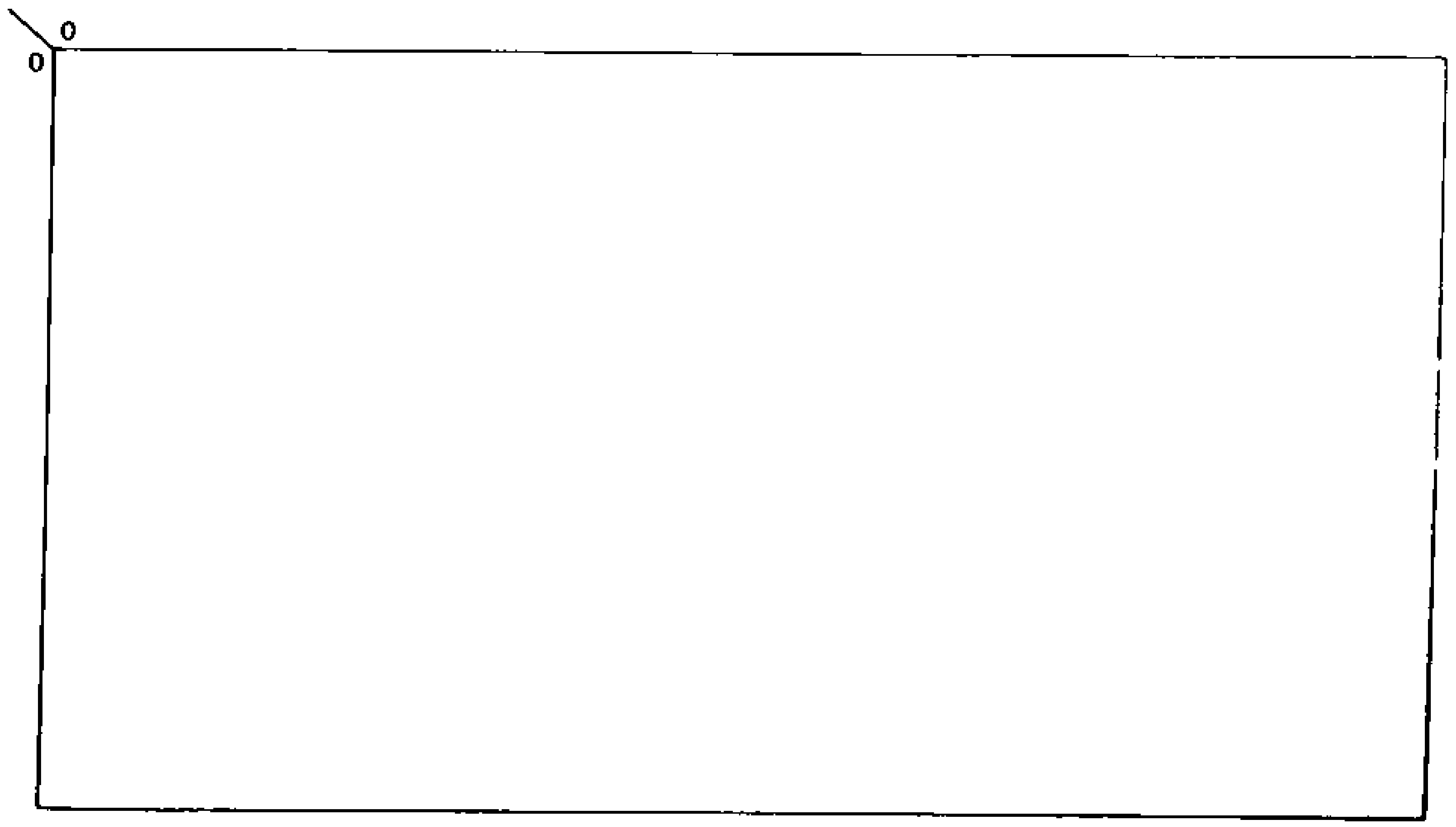
GRAPHICS 1+16+32	is gelijk aan GRAPHICS 1	zonder wissen, zonder tekstvenster
GRAPHICS 7+16+32	is gelijk aan GRAPHICS 7	zonder wissen, zonder tekstvenster
etc.		

Het is niet nodig dat de getallen 16 en 32 expliciet genoemd worden. Voor een betere leesbaarheid van het programma is het echter beter om bijvoorbeeld te spreken van GRAPHICS 1+16, dan GRAPHICS 17.

De GTIA schermen 9, 10 en 11

Het grafische werk van de Atari wordt verzorgd door twee chips: de ANTIC en de GTIA. De ANTIC is eigenlijk een kleine microprocessor, compleet met een instruktieset, een programma en gegevens. Het is deze chip die bijna al het computerwerk voor de grafiek doet. De GTIA 'vertaalt' de digitale informatie van de ANTIC en soms de 6502 processor in een signaal dat naar het beeldscherm gaat. Daarnaast zorgt de GTIA voor de kleuren, de player/missile grafiek en de ontdekkingen van 'botsingen' op het scherm.
Niet in elke Atari zit een GTIA. De oudere Atari-computers hebben een CTIA chip. Dit staat voor Computer Television Interface Adapted integrated circuit (IC voor het aanpassen van computersignaal naar televisiesignaal). Deze chip is in staat alle tot nu toe genoemde schermmodi te behandelen.
Nadat de verschillende Atari's al een tijdje op de markt waren, werd de CTIA een stuk verbeterd. Vanaf dat moment werd hij GTIA genoemd (waarbij er geruchten gaan dat de G nu voor George - de maker van de chip - zou staan).
De GTIA heeft enkele extra truuks, waardoor drie extra grafische schermmodi ter beschikking aan de gebruiker komen. Bijna elke Atari-eigenaar heeft nu de GTIA-chip. Atari heeft een tijdlang deze chip gratis ter beschikking gesteld van de Atari bezitters die de CTIA nog hadden. De drie extra grafische schermmodi zijn alledrie zeer sterke variaties op schermmodus 8.
Geen van de GTIA schermen heeft een tekstvenster. Alledrie hebben ze dezelfde resolutie:

Scherm 9:	80 kolommen bij 192 rijen (geen tekstvenster)
Scherm 10:	80 kolommen bij 192 rijen (geen tekstvenster)
Scherm 11:	80 kolommen bij 192 rijen (geen tekstvenster)



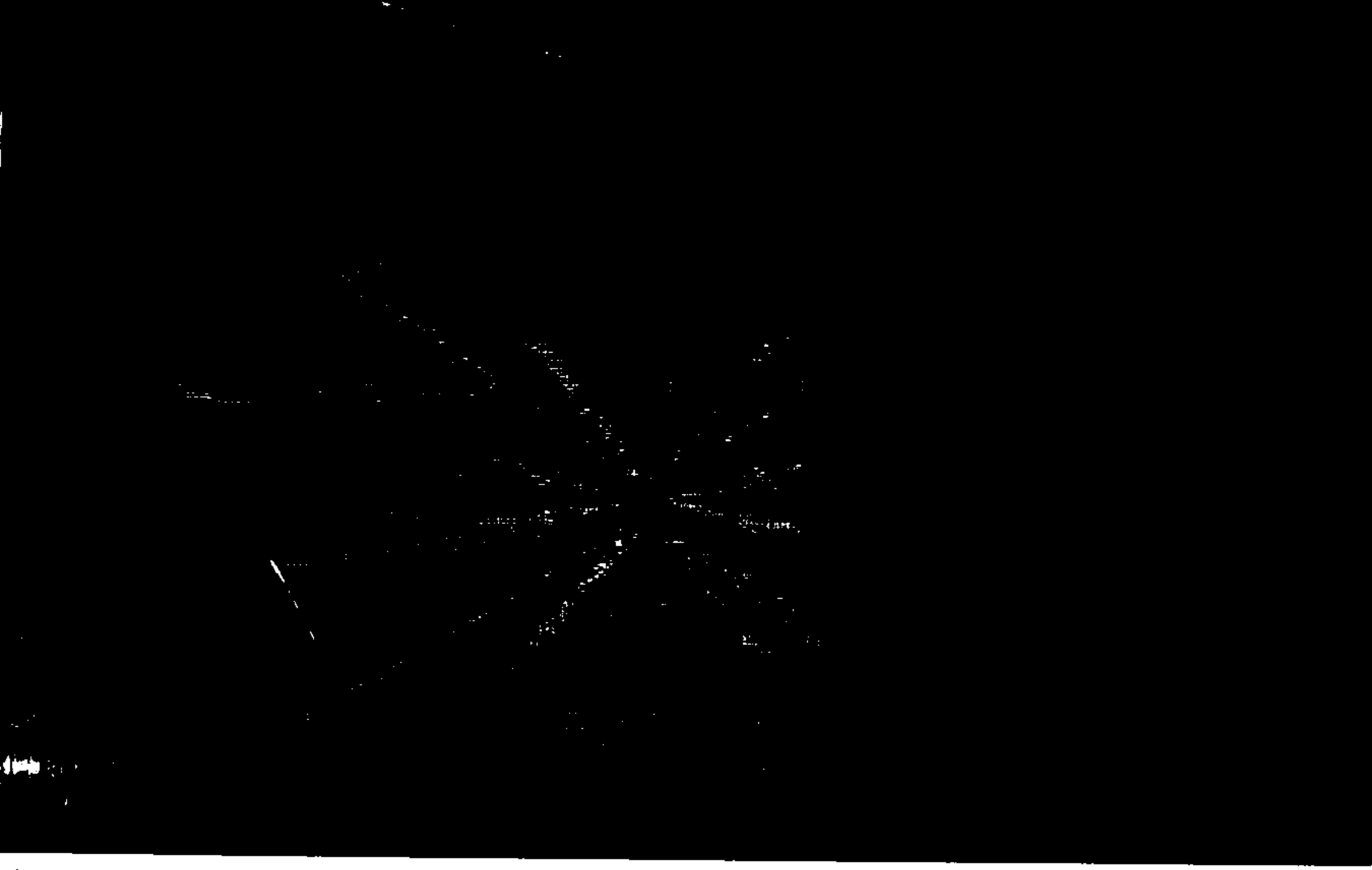
Scherm 9 biedt maar liefst 16 verschillende tinten (helderheden) van één willekeurige kleur. U kunt maar één kleur op het scherm zetten, maar daarvan wel 16 verschillende maten van helderheid.
Scherm 11 biedt 16 verschillende kleuren, allen met dezelfde helderheid. Scherm 10 biedt van allebei wat: 9 verschillende kleuren tegelijk op het scherm, gekozen uit de 128 mogelijke kleuren (16 kleuren, elk met 8 helderheden).

Scherm 9

Dit scherm biedt 16 verschillende tinten van een bepaalde kleur. Daardoor is dit scherm bijzonder geschikt voor het suggereren van dieptewerking.
Tot nu toe heeft u kunnen zien dat de Atari geschikt is voor het weergeven van 128 kleuren, namelijk 16 kleuren met elk 8 helderheden (de helderheid loopt van 0 tot en met 15, maar alleen de even waarden werken). In scherm 9 zijn 16 verschillende helderheden mogelijk, dit gecombineerd met 16 kleuren en geeft u de beschikking over maar liefst 256 kleuren. Deze kleuren zijn weliswaar niet allemaal tegelijk op het scherm te zetten, maar elke kleur is wel individueel aan te roepen.

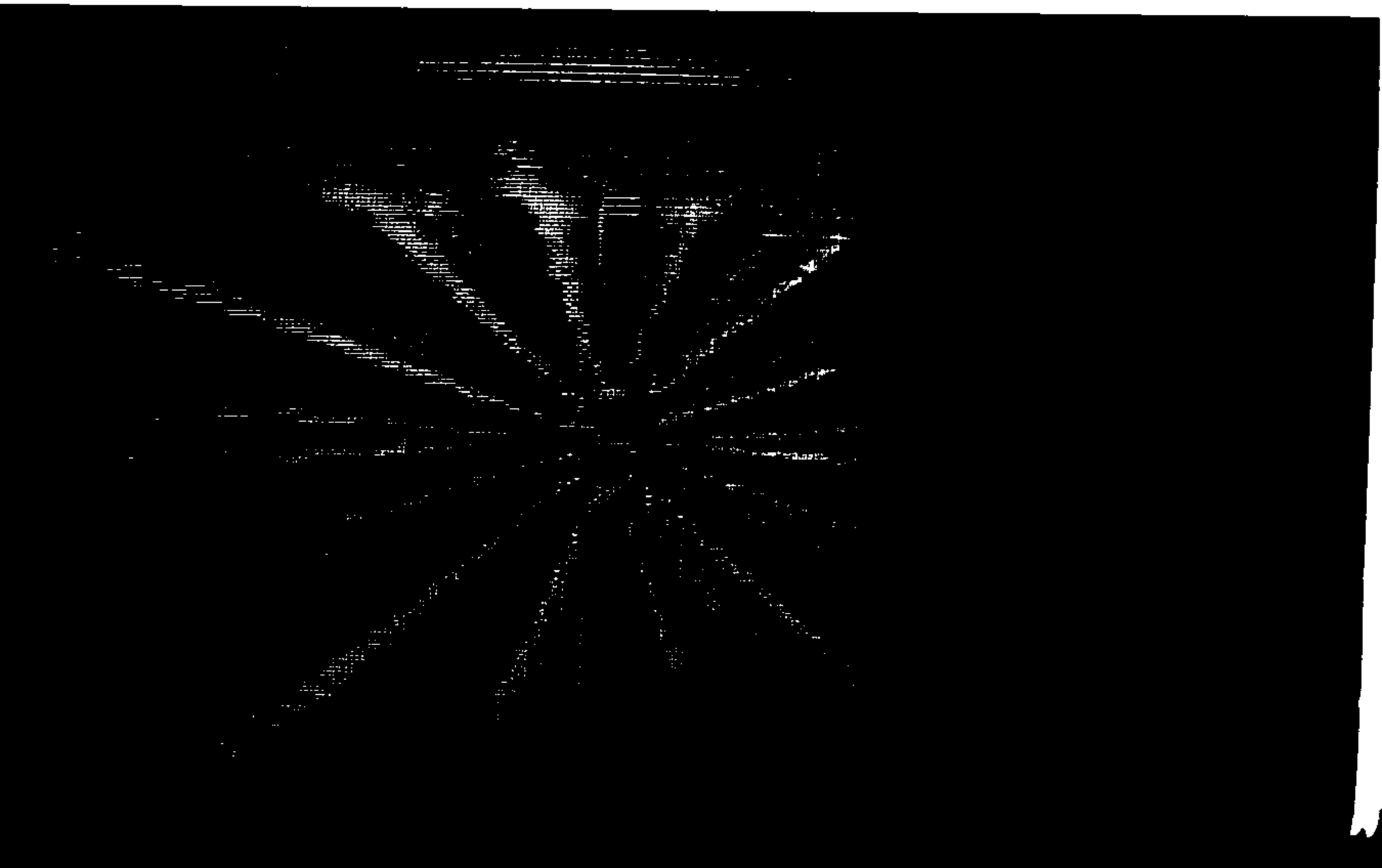
```
10 GRAPHICS 9
20 SETCOLOR 4,7,0
25 FOR C=0 TO 2
30 FOR I=3 TO 15
40 COLOR I
50 PLOT C*26+I-3,C*45
60 DRAWTO C*26+I-3,185
70 NEXT I
80 FOR I=15 TO 3 STEP -1
90 COLOR I
100 PLOT 26-I+C*26,C*45
110 DRAWTO 26-I+C*26,185
120 NEXT I
125 NEXT C
130 GOTO 130
```

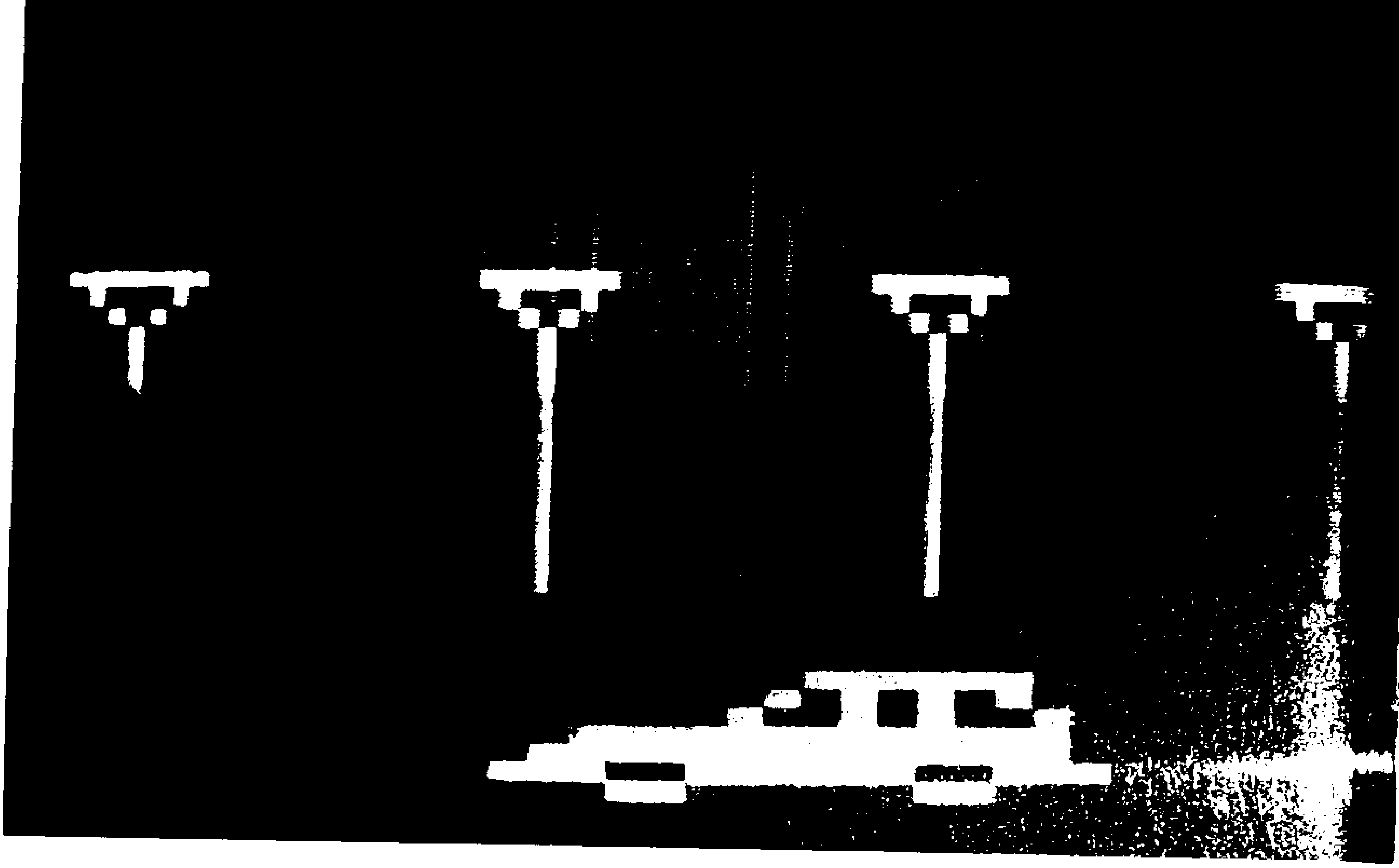
Dit programma laat drie naast elkaar staande cilinders zien. In feite zijn het natuurlijk geen cilinders, maar rechthoeken waarvan de kleur verloopt. Omdat de Atari de kleur zo vloeiend kan laten verlopen, lijkt het net alsof de rechthoeken diepte bezitten.
Regel 10 kiest de schermmodus.
Regel 20 stelt de kleur in. In dit geval donkerblauw, maar u kunt elke kleurwaarde als tweede cijfer achter de SETCOLOR opdracht invullen. Regel 40 kiest een kleur. Omdat we te maken hebben met een één-kleur, zestien helderheden modus, zal de computer telkens een andere tint van de in regel 20 gekozen kleur laten zien.



Graphics kleur programma *blz. 52*

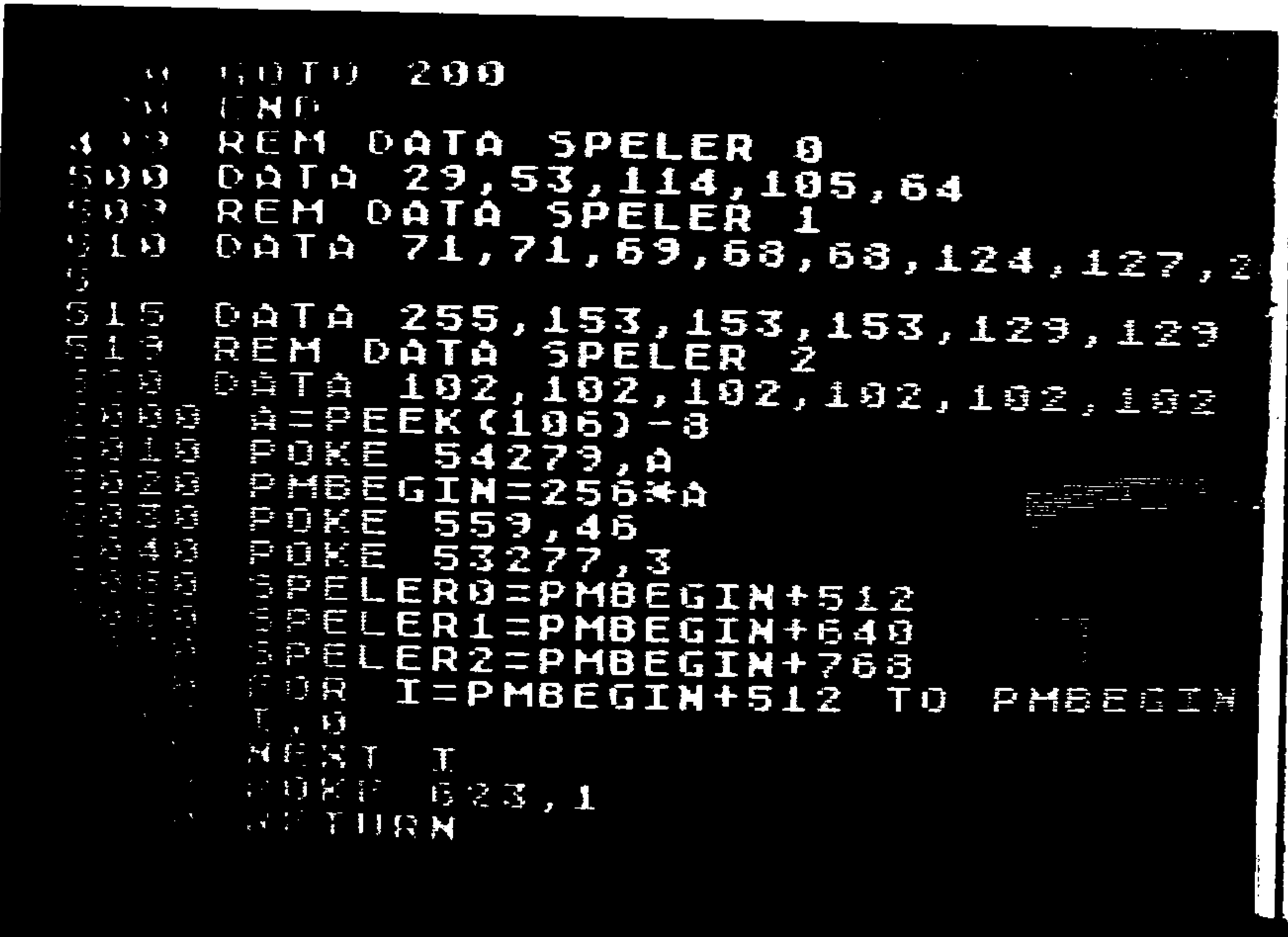
Graphics kleur programma *blz. 52*





Dubbelgroot en beweging programma *blz.* 22[illegible]

Locomotief programma *blz.* 232



De regels 50 tot en met 110 zorgen ervoor dat er telkens een streep getrokken wordt, steeds weer in een iets andere tint.
Om de dieptewerking nog sterker te laten worden, kunt u suggereren dat het licht niet van voren op de cilinders valt, maar een beetje van opzij. Verander regel 30 en 80 in:

```
30 FOR I=5 TO 15
80 FOR I=15 TO 1 STEP -1
```

Het volgende programma geeft een variatie op dit thema. Met dit programma kunt u elke gewenste tekst in 'neon' letters op het scherm zetten. De letters van de tekst worden gevormd door telkens een lijn tussen twee punten te trekken. Als de ene letter klaar is, en met de volgende begonnen moet worden, wordt er geen lijn getrokken, maar een punt geplaatst. Vergelijk dit met de techniek die gebruikt is om de kaart van Nederland te tekenen.

```
10 GRAPHICS 9
20 SETCOLOR 4,13,0
30 FOR HE=0 TO 15
40 COLOR HE
50 FOR GE=1 TO 23
60 READ X,Y
70 IF X=0 AND Y=0 THEN READ X,Y:PLOT X-5+HE/3,Y-15+HE:NEXT GE
80 DRAWTO X-5+HE/3,Y-15+HE
90 NEXT GE
100 RESTORE
110 NEXT HE
120 GOTO 120
130 END
500 DATA 0,0,5,120,5,60,15,120,15,60,0,0
510 DATA 35,60,25,60,25,120,35,120,0,0,25,90,30,90,0,0
520 DATA 48,60,52,60,55,80,55,100,52,120
525 DATA 48,120,45,100,45,80,48,60,0,0
530 DATA 65,120,65,60,75,120,75,60
```

Probeert u zelf of u met de hier aangedragen methoden nog andere dieptewerkingen kunt maken. Probeer eerst om de NEON letters aan twee of meer kanten donker weg te laten lopen.
Een voorproefje van schermmodus 11 kunt u krijgen door regel 10 te veranderen in GRAPHICS 11 en regel 20 te verwijderen.

Scherm 10

Scherm 10 gebruikt 9 van de 128 kleuren die normaal gesproken mogelijk zijn op de Atari. Voor elke kleur wordt een apart 'kleurregister' gebruikt. Deze

kleurregisters vindt u in de geheugenadressen 704 tot en met 712. Dit zijn andere registers dan u tot nu toe gewend was. Het zijn er immers 9 in plaats van 5. Er zijn veel overlappingen, maar door speciale truuks zijn de makers van de Atari er in geslaagd om 9 aparte plaatsen te creeeren waar kleurgegevens opgeslagen kunnen worden.
Elk van deze plaatsen kan onafhankelijk van de anderen gebruikt worden. De kleuren die u kunt kiezen hebben code 0 tot en met 255. Een bepaalde kleur plus helderheid kiest u weer door:

resultaat = 16*kleur + helderheid

Het volgende programma laat een vlak van kleur veranderen door telkens een waarde in een geheugenplaats te POKEn.

```
10 GRAPHICS 10
20 COLOR 0
30 FOR PK=0 TO 255
40 POKE 704,PK
50 FOR WA=1 TO 50:NEXT WA
60 NEXT PK
70 GOTO 70
```

Het programma gebruikt slechts een geheugenplaats, nummer 704, die voor de achtergrondkleur wordt gebruikt. Door hier telkens een kleurwaarde te POKEn, ziet u alle kleuren van licht naar donker verlopen.

Bij de PLOT- en DRAWTO-opdrachten moet u de kleur opgeven door een COLOR opdracht. De kleuren die door de COLOR opdracht gegeven worden zijn afhankelijk van de kleurwaarden die in de verschillende geheugenadressen geplaatst zijn:

POKE 704,	16*kleur+helderheid	bepaalt	COLOR 0 (achtergrond)
POKE 705,	,,	,,	COLOR 1
POKE 706,	,,	,,	COLOR 2
POKE 707,	,,	,,	COLOR 3
POKE 708,	,,	,,	COLOR 4
POKE 709,	,,	,,	COLOR 5
POKE 710,	,,	,,	COLOR 6
POKE 711,	,,	,,	COLOR 7
POKE 712,	,,	,,	COLOR 8

U kunt dit uitproberen door in alle negen adressen een kleur te plaatsen en vervolgens met een COLOR- en DRAWTO-opdracht te kijken of de kleuren kloppen:

```
10 GRAPHICS 10
20 POKE 704,7:REM BLAUW
30 POKE 705,11*16+10:REM BLAUW/GROEN
40 POKE 706,15*16+10:REM LICHT ORANJE
50 POKE 707,5*16+14:REM PAARS (HELDER)
60 POKE 712,12*16+15:REM GROEN (HELDER)
70 COLOR 1:PLOT 1,1:DRAWTO 30,190
80 COLOR 2:PLOT 2,1:DRAWTO 40,180
90 COLOR 3:PLOT 3,1:DRAWTO 50,170
100 COLOR 8:PLOT 5,1:DRAWTO 70,160
110 GOTO 110
```

U ziet dat in de verschillende adressen een kleur geplaatst is. Door met een bepaalde COLOR-opdracht naar het adres te verwijzen, kan die kleur op het scherm worden geplaatst.
Het volgende programma tekent in 8 kleuren een steeds kleiner wordend figuur op het scherm. Vervolgens wordt gebruik gemaakt van de snelheid waarmee het mogelijk is de inhoud van een geheugenplaats te veranderen. Door de inhoud van elk van de kleur-geheugenplaatsen telkens een positie op te schuiven, worden de verschillende COLOR opdrachten telkens voorzien van een andere kleur. Als u een lijn getekend heeft met COLOR 1, dan wordt die lijn op het scherm getoond in de kleur waarvan de waarde in geheugenplaats 705 staat. Als de inhoud van 705 verandert, verandert de kleur van de lijn op het scherm direkt. De lijn hoeft niet opnieuw in een andere kleur getrokken te worden. De kleurwisseling is onmiddellijk.
Door dit effekt te gebruiken, lijkt het net alsof de tekening van dit programma een 'looplicht' heeft.

```
10 GRAPHICS 10
20 FOR A=704 TO 712
30 READ PK
40 POKE A,PK
50 NEXT A
60 FOR P=1 TO 55
70 COLOR KL:KL=KL+1:IF KL=9 THEN KL=1
80 PLOT P,P*1.8
90 DRAWTO P,191-P
100 DRAWTO 79-P,55-P
110 DRAWTO P,P*1.8
120 NEXT P
190 Q=PEEK(712)
200 FOR A=712 TO 706 STEP -1
210 POKE A,PEEK(A-1)
220 NEXT A
230 POKE 705,Q
240 FOR WA=1 TO 20:NEXT WA
250 GOTO 190
500 DATA 97,24,36,45,55,80,114,123,187
```

Regel 70 van dit programma zorgt ervoor dat er alleen zinvolle COLOR opdrachten worden gegeven. De regels 190-230 zorgen ervoor dat de inhoud van elke kleur-geheugenplaats verplaatst wordt naar de volgende kleur-geheugenplaats. Let op het gebruik van de hulpvariabele Q. Deze is vergelijkbaar met de hulpvariabele die in de sorteerprogramma's gebruikt werd. Regel 250 zorgt ervoor dat het 'doordraaien' van de kleurwisseling door blijft gaan. Het programma wordt gestopt door de BREAK of RESET toets.
In regel 500 worden de kleurwaarden opgegeven. U kunt daar zelf uw eigen kleuren invullen. Wilt u bijvoorbeeld alleen verschillende helderheden van een enkele kleur, dan vult u direkt op elkaar volgende waarden in. Om een oranje looplicht te krijgen, verandert u regel 500 in:

```
500 DATA 97,37,38,39,40,41,42,43,44
```

Scherm 11

Dit scherm kan slechts van een enkele helderheid gebruik maken, maar dan wel in 16 kleuren!
De achtergrond is altijd kleur 0 (zwart). De kleuren komen overeen met de 16 standaardkleuren die de Atari kent. Zie hiervoor de tabel eerder in dit hoofdstuk.
De helderheid wordt ingesteld door een SETCOLOR opdracht:

SETCOLOR 4,0,helderheid (van 0 t/m 15)

Het volgende programma geeft een voorbeeld:

```
10 GRAPHICS 11
20 SETCOLOR 4,0,12
30 FOR X=-30 TO 30
40 COLOR ABS(X/2)
50 Y=SQR(900-X*X)
60 PLOT 35+X,100+X*2:DRAWTO 35+Y,100-Y*2
70 PLOT 45+X,100+Y*2:DRAWTO 45+X,100-Y*2
80 NEXT X
90 GOTO 90
```

Als nieuwe opdracht ziet u in regel 40 de opdracht

ABS (= ABSoluut)

Deze opdracht zorgt ervoor dat een getal altijd positief is. In dit programma is dat noodzakelijk, omdat alle kleurcodes positieve getallen zijn. In regel 40 wordt de waarde van X telkens door 2 gedeeld. X loopt van -30 tot +30. Als -30

door twee gedeeld wordt, is het resultaat -15. Om daar 15 uit te krijgen (een toegestane kleurcode) wordt de ABS opdracht gebruikt.

ABS(-15) = 15
ABS(15) = 15

Een andere nieuwe BASIC statement is

SQR

Deze funktie neemt de vierkantswortel van een getal. Een vierkantswortel van een bepaald getal is het getal dat met zichzelf vermenigvuldigd moet worden om het aangegeven getal als resultaat te krijgen. De vierkantswortel van 4 is 2, want 2*2=4. De vierkantswortel van 169 is 13 want 13*13=169.
In dit programma wordt gebruik gemaakt van deze funktie om de vergelijking van een cirkel op te stellen.
Het manipuleren van de PLOT- en DRAWTO-opdrachten in regel 60 en 70 zorgen ervoor dat een cirkel in een verwrongen vorm wordt getekend.

15. GRAFIEK EN TEKST

Hoewel de Atari verschillende schermen kent die specifiek bedoeld zijn voor het maken van grafische afbeeldingen, is het soms handig om op een tekstscherm (scherm 0) tekeningen te kunnen maken. Het kan voorkomen dat u tekst met een afbeelding wilt verluchtigen.
Speciaal voor scherm 0 (en dus ook voor de tekstvensters) zijn de Atari computers uitgerust met een schat aan grafische symbolen. Zie voor alle beschikbare grafische symbolen de ***ASCII code tabel in de bijlagen***.
Door de symbolen bij elkaar op het scherm te plaatsen, kunt u zelfs tekeningen maken.
Aan het maken van een boek over een computer, is bij de hoofdstukken over grafiek altijd het nadeel verbonden, dat de precisie en scherpte waarmee een computer grafische symbolen op het scherm plaatst, niet of nauwelijks op papier te evenaren is. Om onduidelijkheden bij het intikken van de programmalistings te voorkomen, zult u in dit hoofdstuk alle grafische tekens in de programmalistings tegenkomen als een PRINT CHR$(..) opdracht. Deze opdracht zorgt er voor dat de computer het symbool afdrukt, dat hoort bij de tussen haakjes geplaatste ASCII-code.
Bij het intikken kunt u dit overnemen, of u kunt de grafische tekens intikken door *gelijktijdig* de CONTROL- en een lettertoets in te drukken.

Het volgende programma geeft een voorbeeld van een eenvoudige tekening op scherm 0.

```
10 GRAPHICS 0
20 PRINT "  ";CHR$(6);CHR$(7)
30 PRINT "  ";CHR$(22);CHR$(2)
40 PRINT "  ";CHR$(22);CHR$(2)
50 PRINT "  ";CHR$(22);CHR$(2)
60 PRINT "  ";CHR$(22);CHR$(2)
70 PRINT "  ";CHR$(22);CHR$(2)
80 PRINT " ";CHR$(6);CHR$(19);CHR$(19);CHR$(7)
```

Met dit programma tekent u een kleine raket bovenaan het scherm.
Regel 10 wist het scherm.
De regels 20 - 80 drukken telkens een paar spaties af, gevolgd door een stel grafische symbolen. Door de juiste grafische symbolen te kiezen, is een willekeurig object samen te stellen.
Door het toevoegen van een POSITION opdracht is het mogelijk de raket op de grond te plaatsen. Voeg de volgende twee programmaregels toe:

```
15 POSITION 2,10
90 FOR X=1 TO 15 : PRINT CHR$(13);:NEXT X
```

Regel 15 zorgt ervoor dat de tekstcursor zich naar beneden begeeft.
Regel 90 tekent de grond.

Lift-off

Om de raket te laten vertrekken is beweging nodig. In dit geval kunnen we de beweging op het scherm maken door de raket telkens opnieuw op het beeldscherm af te drukken. Elke keer wordt de raket een beetje verschoven ten opzichte van de vorige keer. In de lage resolutie van het tekstscherm wil dat zeggen dat de raket telkens een regel hoger afgedrukt wordt. Probeert u het vol gende programma.

```
10 GRAPHICS 0
20 FOR Y=13 TO 1 STEP -1
30 POSITION 2,Y
40 PRINT "  ";CHR$(6);CHR$(7)
50 PRINT "  ";CHR$(22);CHR$(2)
60 PRINT "  ";CHR$(22);CHR$(2)
70 PRINT "  ";CHR$(22);CHR$(2)
80 PRINT "  ";CHR$(22);CHR$(2)
90 PRINT "  ";CHR$(22);CHR$(2)
100 PRINT " ";CHR$(6);CHR$(19);CHR$(19);CHR$(7)
110 PRINT "      "
120 POSITION 2,21
130 FOR X=1 TO 15:PRINT CHR$(13);:NEXT X
140 NEXT Y
```

Dit programma lijkt erg veel op het vorige programma. In dit programma wordt de raket echter vele malen getekend, telkens op een andere plaats.
Regel 10 zorgt voor het scherm.
Regel 20 maakt een lus die bepaalt waar de raket getekend gaat worden.
Regel 30 is de positionering van de tekstcursor.
De regels 40-100 tekenen de raket. Regel 110 tekent een blanko streep. Waarvoor is deze nodig? Omdat de raket telkens een positie hoger op het scherm getekend wordt, wordt bijna de hele raket gewist door de nieuw te tekenen raket. Alleen de onderste regel van de tekening wordt niet uitgewist. Om te zorgen dat de raket geen 'staart' krijgt, moet de onderste regel telkens gewist worden door een blanko regel. Probeer zelf uit hoe het eruit ziet als regel 110 verwijderd wordt. Het resultaat is meer een haarborstel dan een opstijgende raket.
Regel 120 zorgt voor een vaste positionering. De grond moet immers, ongeacht de plaats van de raket, telkens op dezelfde positie getekend worden.
Regel 140 sluit de lus af.

Hinderlijk aan bovenstaand programma is het geknipper van de cursor. Deze geeft aan waar er op het scherm geschreven gaat worden. Doordat dit telkens een ander punt is, knippert de cursor overal over het scherm heen. De Atari kent een geheugenadres waarin informatie over de tekstcursor staat. Als in geheugenadres 752 een 0 staat, wordt de tekstcursor op het scherm afgebeeld. Bevat dit geheugenadres de waarde 1, dan wordt de juiste plaats voor het

afdrukken wel afgedrukt, maar de cursor zelf wordt niet geplaatst. U kunt zelf deze 1 in die geheugenlokatie plaatsen met een POKE opdracht:

POKE 752,1 (cursor uit)

Plaats deze opdracht in een nieuwe regel 15, en bekijk het resultaat. Veel rustiger, niet?

Na afloop van het programma verschijnt de tekst READY onderin het beeld. Een beetje hinderlijk, want zo er al bij het lanceren van raketten over READY gesproken wordt, dan is dat voor het vertrek, en zeker niet erna. Een truuk hiervoor heeft u echter al in het inleidende hoofdstuk over grafieken kunnen lezen. Het programma mag niet aflopen, daarom plaatst u aan het einde van het programma een oneindige lus. Voeg toe:

```
150 GOTO 150
```

Om de raket wat echter te doen lijken is in het volgende programma de vuurmond opgenomen en is de raket van geluid voorzien. De raket blijft op de grond totdat u de S toets van 'start' indrukt.

```
5 DIM A$(1)
10 GRAPHICS 0
20 POKE 752,1
30 Y=14
40 GOSUB 390
50 SOUND 0,50,0,7
60 OPEN #1,4,0,"K:"
70 GET #1,A
80 IF CHR$(A)<>"S" THEN GOTO 70
90 FOR Y=13 TO 1 STEP -1
100 GOSUB 390
110 NEXT Y
120 SOUND 0,0,0,0:GOTO 120
130 END
390 POSITION 2,Y
400 PRINT "  ";CHR$(6);CHR$(7)
410 PRINT "  ";CHR$(22);CHR$(2)
420 PRINT "  ";CHR$(22);CHR$(2)
430 PRINT "  ";CHR$(22);CHR$(2)
440 PRINT "  ";CHR$(22);CHR$(2)
450 PRINT "  ";CHR$(22);CHR$(2)
460 PRINT " ";CHR$(6);CHR$(19);CHR$(19);CHR$(7)
470 PRINT "  ";CHR$(35);CHR$(35)
480 SOUND 0,Y*Y,14,Y+1
490 FOR WA=1 TO 14:NEXT WA
500 POSITION 2,Y+7
510 PRINT "     "
```

```
520 POSITION 2,21
530 FOR X=1 TO 15:PRINT CHR$(13);:NEXT X
540 RETURN
```

In dit programma is gebruik gemaakt van een subroutine om het veelvuldig intikken van alle PRINT opdrachten te voorkomen.
Nieuw zijn de opdrachten in de regels 60-80.
In regel 60 wordt een IOCB kanaal geopend. Voor een uitgebreide uitleg van de IOCB wordt u verwezen naar het hoofdstuk *'Bestanden verwerken'* en naar de *bijlages*. In dit programma wordt kanaal 1 (vrij voor diverse doeleinden) geopedn met als taak de invoer van gegevens (de 4). Bij gebruik van het toetsenbord moet het derde getal achter de OPEN opdracht een nul zijn. De OPEN opdracht wordt afgesloten met een K: code om aan te geven dat de computer invoer van het toetsenbord kan verwachten.
Regel 70 neemt invoer van het toetsenbord. Als de gebruiker een toets indrukt, wordt de ASCII code van de ingetoetste letter opgeslagen in de A achter de GET opdracht.
Regel 80 controleert of de ingedrukte toets een S is. Zo niet dan wordt terug gesprongen naar regel 70 voor de invoer van een nieuwe toetsindruk.
Het voordeel van deze methode boven de INPUT opdracht, is dat de computer geen vraagteken op het scherm zet, en dat de gebruiker de RETURN toets niet hoeft in te drukken. Zodra er een toets ingedrukt wordt, reageert de GET opdracht en plaatst de waarde in de genoemde variabele.
Regel 470 tekent telkens heel even een paar 'matjes' onder de raket als een soort vuurmond. De matjes worden uitgewist door de spaties van regel 510.
De SOUND opdrachten komen uitgebreid ter sprake in de hoofdstukken over geluid en muziek.

Lucifers

Het volgende programma laat zien dat ook met alleen lage resolutie grafiek best leuke programma's te maken zijn. Het spel zelf is heel eenvoudig. Door echter veel 'aankleding' te gebruiken, wordt het voor de speler toch een aantrekkelijk geheel.

```
10 REM luciferspel
20 GRAPHICS 0
25 POKE 752,1
30 PRINT :PRINT :PRINT
40 PRINT "*************************************"
50 PRINT :PRINT
60 PRINT "         HET 23 LUCIFERS SPEL "
70 PRINT :PRINT "Elke beurt mag u 1,2 of 3 lucifers"
80 PRINT "weghalen. Ik doe hetzelfde."
90 PRINT "De speler die de laatste lucifer moet"
100 PRINT "weghalen, heeft verloren."
```

```
110 PRINT :PRINT
120 PRINT "**************************************"
130 PRINT :FOR WA=1 TO 1500:NEXT WA
140 PRINT "WILLEKEURIG gekozen mag ...."
150 LU=23
160 FOR A=1 TO 10:PRINT CHR$(127);CHR$(253);
165 FOR WA=1 TO 100:NEXT WA:NEXT A
170 FOR WA=1 TO 600:NEXT WA
180 IF RND(0)<0.5 THEN GOTO 300:REM SPELER BEGINT
190 GRAPHICS 0
200 PRINT
210 PRINT "IK beginnen."
220 PRINT
230 PRINT "Ik haal 2 lucifers weg."
240 LU=21:GOSUB 1000:REM WEERGAVE LUC.
250 GOTO 500:REM SPEL-LUS
299 REM SPELER
300 GRAPHICS 0
310 PRINT :PRINT
320 PRINT "U beginnen."
330 LU=23:GOSUB 1000:REM WEERGAVE LUC.
499 REM BEGIN SPEL-LUS
500 PRINT :PRINT :PRINT "Hoeveel lucifers neemt u weg?"
510 PRINT "(1,2 of 3)";:INPUT SP
520 IF SP=1 OR SP=2 OR SP=3 THEN LU=LU-SP:GOTO 530
525 PRINT "U SPEELT VALS!!!":GOTO 500
530 GRAPHICS 0:GOSUB 1000:REM WEERGAVE LUC.
540 PRINT :PRINT :PRINT "Ik denk ......."
550 FOR Z=1 TO 10:PRINT CHR$(127);CHR$(253);
555 FOR WA=1 TO 30:NEXT WA:NEXT Z
560 IF LU=4 THEN MS=3:GOSUB 2000:GOTO 500
570 IF LU=3 THEN MS=1:GOSUB 2000:GOTO 500
580 IF LU=2 THEN MS=1:GOSUB 2000:GOTO 500
590 IF LU=1 THEN GOTO 700:REM GEWONNEN
600 IF LU<=0 THEN GOTO 900:REM VERLOREN
610 IF LU>4 THEN MS=4-SP:GOSUB 2000:GOTO 500
620 PRINT "FOUT IN PROGRAMMALOOP"
630 END
699 REM SPELER GEWONNEN
700 GRAPHICS 0
710 POSITION 2,2
720 FOR LR=1 TO 36
730 PRINT CHR$(20);
740 NEXT LR
750 FOR BB=3 TO 22
760 POSITION 2,BB
770 FOR LR=1 TO 36
```

```
780 PRINT CHR$(25);
790 NEXT LR
800 PRINT CHR$(127);CHR$(253):NEXT BB
830 POSITION 6,12:PRINT "U HEEFT GEWONNEN!!"
840 END
899 REM SPELER VERLOREN
900 GRAPHICS 0
910 POSITION 6,12:PRINT "JAMMER, U HEEFT VERLOREN."
920 END
999 REM WEERGAVE LUCIFERS
1000 PRINT
1010 PRINT "     Nog ";LU;" lucifers aanwezig."
1020 POSITION 6,6
1030 FOR LR=1 TO LU
1040 PRINT CHR$(20);
1050 NEXT LR
1060 FOR BB=7 TO 14
1070 POSITION 6,BB
1080 FOR LR=1 TO LU
1090 PRINT CHR$(25);
1100 NEXT LR:NEXT BB
1110 RETURN
2000 GRAPHICS 0
2010 PRINT "Ik neem ";MS;" lucifers weg."
2020 LU=LU-MS
2030 GOSUB 1000:REM WEERGAVE LUC.
2040 GOTO 500:REM SPEOL-LUS
2050 RETURN
```

De regels 10-130 zorgen voor een introduktiebladzijde. De speler krijgt te zien om welk spel het gaat en hoe de spelregels zijn.
De regels 140-170 simuleren dat de computer aan het denken is. In werkelijkheid denkt de computer natuurlijk vele malen sneller. Door hier opzettelijk een ruime vertraging in te bouwen, wordt de speler tevens de kans geboden de spelregels nogmaals te lezen. Een andere methode hiervoor heeft u al in vorige programma's gezien: de speler moet op een toets drukken als hij/zij klaar is met lezen. Regel 160 zorgt voor 10 piepjes. De WA lus zorgt voor een korte pauze tussen de piepjes in, en een lange pauze na de piepjes.
Regel 180 zorgt ervoor dat niet telkens dezelfde speler mag beginnen. Door een willekeurig getal te nemen met een RND opdracht, krijgt de computer een waarde tussen 0 en 1 (nooit 1). Door deze waarde te vergelijken met .5 zal de computer gemiddeld een op de twee keer de vergelijking als waar beschouwen en naar regel 300 springen. Als de computer naar regel 300 springt, dan begint de speler. Gaat de computer verder met regel 190, dan begint de computer.

De regels 200-250 geven aan dat de computer begint door twee lucifers weg te halen.

Regel 240 geeft het nieuwe aantal lucifers aan (23-2=21). Vervolgens wordt naar de subroutine die begint op regel 1000 gesprongen.
Deze subroutine zorgt elke keer opnieuw voor de weergave van het juiste aantal lucifers.
Regel 250 laat de computer het programmadeel, waarin behandeld wordt dat de speler begint, overslaan. Op regel 500 begint de lus die de rest van het spel regelt.
De regels 300-330 zijn gelijk aan 220-250, maar in dit geval begint de speler. Op regel 500 begint de spel-lus. Hier zal de computer net zolang terug komen tot het spel beeindigd is.
De regels 500-525 vragen de speler om aan te geven hoeveel lucifers hij/zij weg wil nemen.
Let op regel 525 die vals spelen verhindert. Regel 530 zorgt, door het aanroepen van de subroutine, op 1000 voor het weergeven van de lucifers. De regels 540 en 550 zorgen voor een vertraging en opnieuw een simulatie van een 'denkende' computer.
De regels 560-610 bepalen hoeveel lucifers de computer telkens wegneemt. Om de regels niet te lang te maken, is het deel dat telkens hetzelfde is, in een subroutine geplaatst (beginnend op regel 2000).
Als regel 610 bereikt wordt, is het spel nog niet in het laatste stadium en gebruikt de computer een eenvoudige aftreksom om te berekenen hoeveel lucifers er weg genomen moeten worden.

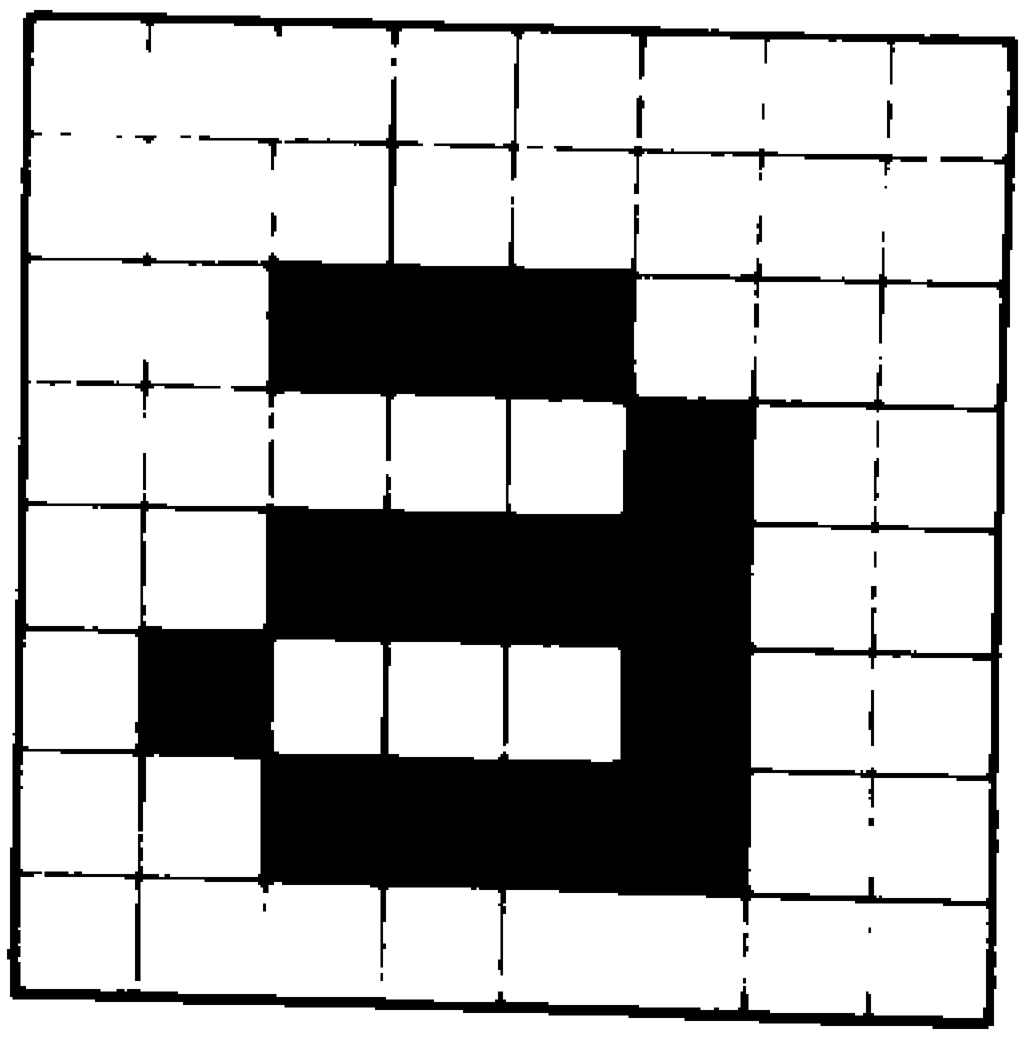
Regel 620 is een extra regel die alleen wordt gebruikt in het test-stadium van een programma. Als het programma goed loopt, kunt u deze regel weghalen. Deze regel geeft aan dat de computer die regel nooit mag bereiken. Doet hij dat toch, dan wordt er op het scherm afgedrukt dat er ergens een fout in de programmaloop zit. Heeft u een programma gemaakt, dat, ondanks het gebruik van een stroomdiagram, niet goed wil lopen en waarvan u vermoedt dat de fout in de programmaloop zit, plaatst u dan op verschillende punten waarvan u zeker bent dat de computer er niet mag komen, dit soort regels. U kunt elke PRINT opdracht ook nog voorzien van een regelaanduiding. Hierdoor zijn de fouten helemaal snel op te zoeken.
De regels 700-840 geven met een feestelijk scherm aan dat de speler gewonnen heeft.
De regels 900-920 geven een sobere melding dat de speler verloren heeft. Wilt u de speler de mogelijkheid bieden om het spel nog een keer te spelen zonder dat hij RUN hoeft in te tikken, dan kunt u tussen regel 920 en 990 een INPUT opdracht plaatsen die de gebruiker vraagt of hij nog een spel wil spelen. Naar aanleiding van het antwoord wordt het programma beeindigd of wordt er terug gesprongen naar het begin. Deze regels moeten natuurlijk voor een END opdracht geplaatst worden.
Om te zorgen dat de keuze mogelijkheid ook bestaat als de speler gewonnen heeft, moet er voor regel 840 een verwijzing geplaatst worden.
De regels 1000-1110 vormen de subroutine die de weergave van de lucifers verzorgt. Eerst wordt het aantal lucifers in letters en cijfers weergegeven (regel 1010). Vervolgens wordt het juiste aantal luciferkoppen getekend (1030-1050). Tot slot worden een aantal malen net zoveel streepje getrokken als er

lucifers zijn. Doordat de streepjes onder elkaar staan, ontstaan er lucifers (1060-1100).
De regels 2000-2050 zorgen voor het rekenwerk van de computer en de weergave van zijn beslissing.

Zelf lettertekens maken

Tot nu toe zijn de figuurtjes in schermmodus 0 gemaakt met behulp van de grafische tekens van de Atari. Maar soms kan het handig zijn om andere figuurtjes ter beschikking te hebben. Zo heeft de Atari een groot aantal verschillende lettertekens (128 normaal, 128 inverse en de internationale lettertekenset), maar als u bijvoorbeeld russische of japanse tekens wilt (die uiteraard niet standaard in een amerikaanse machine zitten), zult u zelf aan de slag moeten. Het is mogelijk om elk gewenst teken zodanig in de Atari te plaatsen, dat het indrukken van een toets dit letterteken op het scherm zet. Op deze wijze kunt u bijna onbeperkt verschillende lettertekens maken. Het is echter praktischer om u tot een maximum van 128 tekens te beperken.

De lettertekens van de Atari bestaan uit allemaal kleine lichtpuntjes. Dit zijn dezelfde lichtpuntjes als waarmee de lijnen van de grafische schermen zijn opgebouwd. Als de gebruiker door middel van een PRINT CHR$ opdracht of door het indrukken van een toets, een bepaald letterteken op het scherm wil zetten, zoekt de computer de opgegeven ASCII code op in een tabel en drukt het patroon van lichtpuntjes dat erbij hoort af. Voor een goed begrip moeten we nu even technisch in het geheugen van de Atari gaan wroeten.
De patronen (ook wel bit-patronen genoemd, omdat ze verdeeld zijn in rijtjes van **8 nullen en enen**) zijn opgeslagen in het ROM gedeelte van het geheugen en starten op geheugenadres 57344. In totaal zijn er 256 normale lettertekens (gewoon + inverse). Elk letterteken is opgebouwd uit 8 rijen van elk 8 bits. Zo'n rooster van 8x8 is voor elk letterteken hetzelfde. Welke lichtpuntjes op moeten lichten en welke niet, is voor elk letterteken anders. Een 'a' ziet er als rooster zo uit:



U ziet dat er flink veel wit om de letter heen zit. De 'a' had groter getekend kunnen worden en zou dan toch nog in het rooster van 8x8 gepast hebben. Dit is niet gebeurd om te zorgen dat de letters niet tegen elkaar aan leunen als de verschillende letterroosters achter elkaar gezet worden om zodoende een regel tekst te vormen. Daarbij komt dat op deze wijze de hoofdletters groter dan de kleine letters afgebeeld kunnen worden.
Elk letterteken is samengesteld uit 8 bytes (een byte is 8 bits).
Zoals al gezegd, zijn de lettertekenpatronen opgeslagen in ROM. Dit is logisch, omdat de Atari telkens na het aanzetten, moet weten hoe de verschillende letters er uit moeten zien. Hij mag dit nooit 'vergeten', en dus is de nodige informatie opgeslagen in dat deel van het geheugen dat niet gewist kan worden. Daar zit dan ook een probleem voor de gebruiker. Het ROM gedeelte van het geheugen is goed beschermd tegen vergissingen van de gebruiker. Het is dan ook onmogelijk de waarden in de verschillende geheugenadressen in ROM door middel van POKE te veranderen.
Het is wel mogelijk om de informatie uit het ROM gedeelte te lezen. Dat gaat door middel van de PEEK opdracht. Als we nu eerst alle informatie uit het ROM gedeelte lezen en in het RAM geheugen plaatsen, kunnen we dat RAM gedeelte gebruiken om te gaan rommelen in de gegevens van de lettertekens. Het RAM geheugen is immers vrij toegankelijk voor de gebruiker. Daar kan de inhoud van de geheugenadressen wel door middel van POKE opdrachten veranderd worden.
Nog twee problemen resten:

1) Hoe weet de computer dat hij de gegevens over de lettertekens uit onze nieuwe lijst in het RAM gebied moet halen in plaats van uit het ROM gebied?
2) Het RAM gebied wordt ook gebruikt voor bijvoorbeeld onze BASIC programma's.

Hoe zorgen we ervoor dat zo'n BASIC programma niet gebruik gaat maken van de geheugenadressen waarin onze lijst van nieuwe lettertekens staat, zodat deze vernietigd wordt?

Het tweede probleem vraagt het minste werk. In geheugenlokatie 106 staat een getal dat aan de Atari opgeeft waar de bovenkant van het geheugen is. Door de Atari te foppen en te zeggen dat de bovenkant van het RAM gebied (de RAMtop) ietsje lager is, hebben we boven de RAMtop nog een stukje geheugen vrij, waar de computer verder niet meer komt. Daar kunnen we mooi de gegevens van de lettertekens zetten.

```
10 POKE 106,PEEK(106)-4 : GRAPHICS 0
```

In geheugenlokatie 106 zetten we de oude waarde van deze lokatie min vier. Daardoor krijgen we 4*256 bytes geheugenruimte vrij. (Waarom een 4 zorgt voor 4*256 bytes geheugenruimte laten we hier maar in het midden. Het zou te ver voeren om alle achterliggende technieken die hiervoor nodig zijn, uit te

leggen. U vindt hier slechts die uitleg die nodig is voor het zelf maken en gebruiken van nieuwe lettertekens.)
De computer denkt nu dat de bovenkant van het geheugen lager is dan die in werkelijkheid is. De GRAPHICS opdracht moet direkt achter het verschuiven van de RAMtop geplaatst worden om te zorgen dat er later in het programma geen vreemde effecten met scherm 0 plaatsvinden.

Het verplaatsen van de lijst van letterteken-gegevens is niet zo moeilijk. We gebruiken een FOR..NEXT lus om telkens een geheugenplaats in het ROM te lezen en deze in het RAM te plaatsen. Voor elk letterteken moeten 8 geheugenplaatsen (8 bytes) worden gekopieerd. Ook bij dit onderdeel zullen we niet ingaan op de technieken die nodig zijn om dit te doen.

```
30 BEGINLIJST=PEEK(106)*256
40 POKE 756,BEGINLIJST/256
50 FOR KOPIE = 0 TO 1023
60 POKE BEGINLIJST+KOPIE, PEEK(57344+KOPIE)
70 NEXT KOPIE
```

In regel 30 wordt gekeken waar het begin van de lijst komt te staan.
In regel 40 wordt de wijzer naar het begin van de nieuwe lijst geplaatst. Zodoende springt de computer, als hij een teken op het scherm wil plaatsen, niet naar de gegevenslijst in ROM, maar naar de nieuwe gegevenslijst in RAM. Op dit punt in het programma is die lijst nog leeg. Het kopieren vindt plaats in de regel 50-70.
In regel 50 wordt de lus ingesteld. Er moet 1024 keer worden gekopieerd. In regel 60 wordt op de juiste plaats (BEGINLIJST+KOPIE) de juiste waarde (57344+KOPIE) gekopieerd.
Regel 70 sluit de lus af.
Als u dit programma runt, wordt het scherm gewist (regel 10) en gebeurt er een tijdje schijnbaar niets. In werkelijkheid is de computer bezig met het kopieren van de gegevens in 1024 geheugenplaatsen. Zodra dit klaar is, geeft de computer de melding READY.
Om het hele kopieerproces zichtbaar te maken, kunt u een extra regel 20 toevoegen. Het programma wordt dan:

```
10 POKE 106,PEEK(106)-4:GRAPHICS 0
20 FOR A=1 TO 124:PRINT CHR$(A);:NEXT A
30 BEGINLIJST=PEEK(106)*256
40 POKE 756,BEGINLIJST/256
50 FOR KOPIE=0 TO 1023
60 POKE BEGINLIJST+KOPIE,PEEK(57344+KOPIE)
70 NEXT KOPIE
```

In regel 20 drukt de computer nadat het scherm ingesteld is, 124 tekens af. De opbouw van deze tekens wordt door de computer afgelezen in de normale lijst met gegevens die in het ROM geplaatst is. Maar in regel 40 verandert het programma de wijzer naar deze lijst. De computer gaat nu de gegevens van de

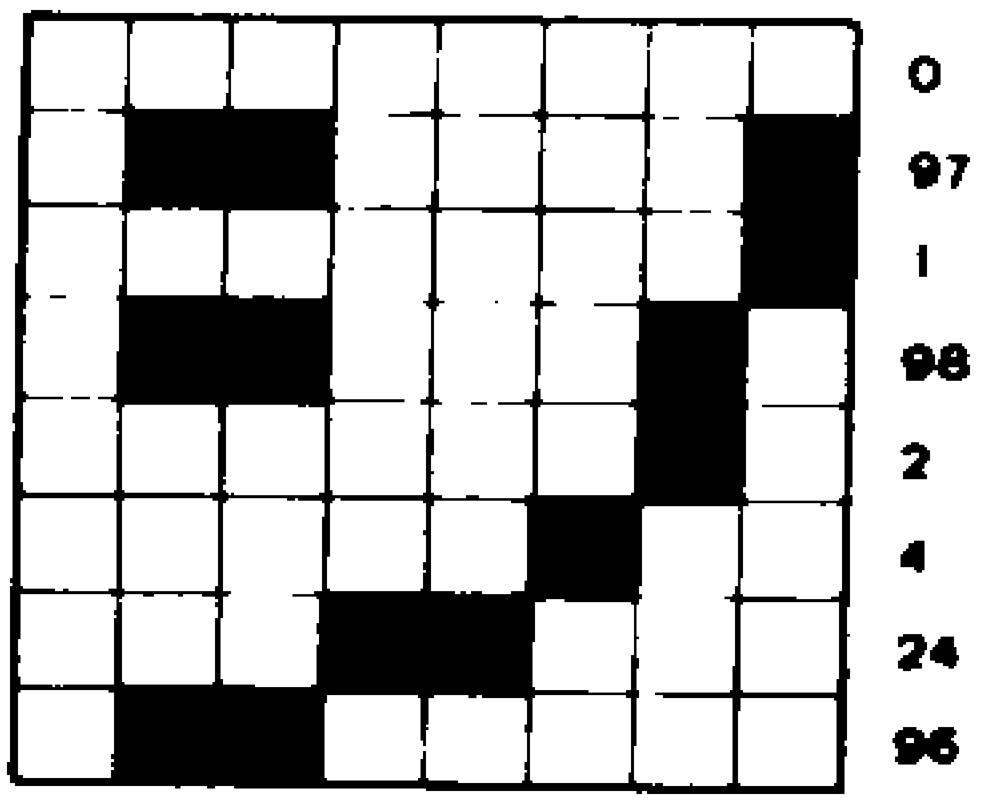
lettertekens in de nieuwe lijst opzoeken. Omdat daar nog niets staat, geeft de computer vreemde tekens, of helemaal niets weer: de letters op het scherm verdwijnen.
Gelukkig wordt de oorspronkelijk lijst uit de ROM, door de regels 40-70 in de nieuwe lijst in RAM gekopieerd. Bij elk teken dat gekopieerd wordt, krijgt de computer meer informatie in de lijst, en kan hij er weer een letterteken bij maken. Het lijstje met tekens wordt langzaam weer compleet. Run dit programma en aanschouw het resultaat.
Onthoud dat na dit programma de originele lijst met lettertekengegevens begint op geheugenlokatie **57344**. De lijst met gekopieerde gegevens, waar we nu verder mee gaan werken, begint op geheugenlokatie **BEGINLIJST**.

Het samenstellen van lettertekens

Nu duidelijk is hoe we een lijst met gegevens over de lettertekens in het RAM gebied kunnen krijgen, is de volgende stap om te kijken wat er veranderd moet worden om een nieuw letterteken te maken.
Zelf-maak lettertekens zijn heel handig als u symbolen uit een ander alphabet wilt gebruiken. Bijvoorbeeld als u uw Atari wilt gebruiken bij uw studie Japans.
Een andere nuttige toepassing is het weergeven van hele kleine tekeningen die u vaak moet gebruiken. In een van de vorige hoofdstukken heeft u het programma 'hengelen' gezien. In dat programma is gebruik gemaakt van standaard-tekentjes. Het is natuurlijk veel leuker als we speciale tekens kunnen gebruiken.
Van elk van deze twee toepassingen volgt nu een voorbeeld om het maken en het gebruik van zelf-maak letterteken duidelijk te maken.

Japans

Japans kent verschillende schriften. Het eenvoudigste is een schrift met een 'alphabet' van zo'n 50 'letters' te leren. Eigenlijk is het geen echt alphabet, want elk teken staat voor een lettergreep en niet voor een letter. Dit schrift heet Katakana. Uit die 50 lettertekens zijn in het komende programma alleen de tekens genomen die nodig zijn voor het schrijven van 'computer' in het Japans.
Als voorbeeld het Japanse letterteken 'SI'. Dit teken ziet er (vereenvoudigd) zo uit:



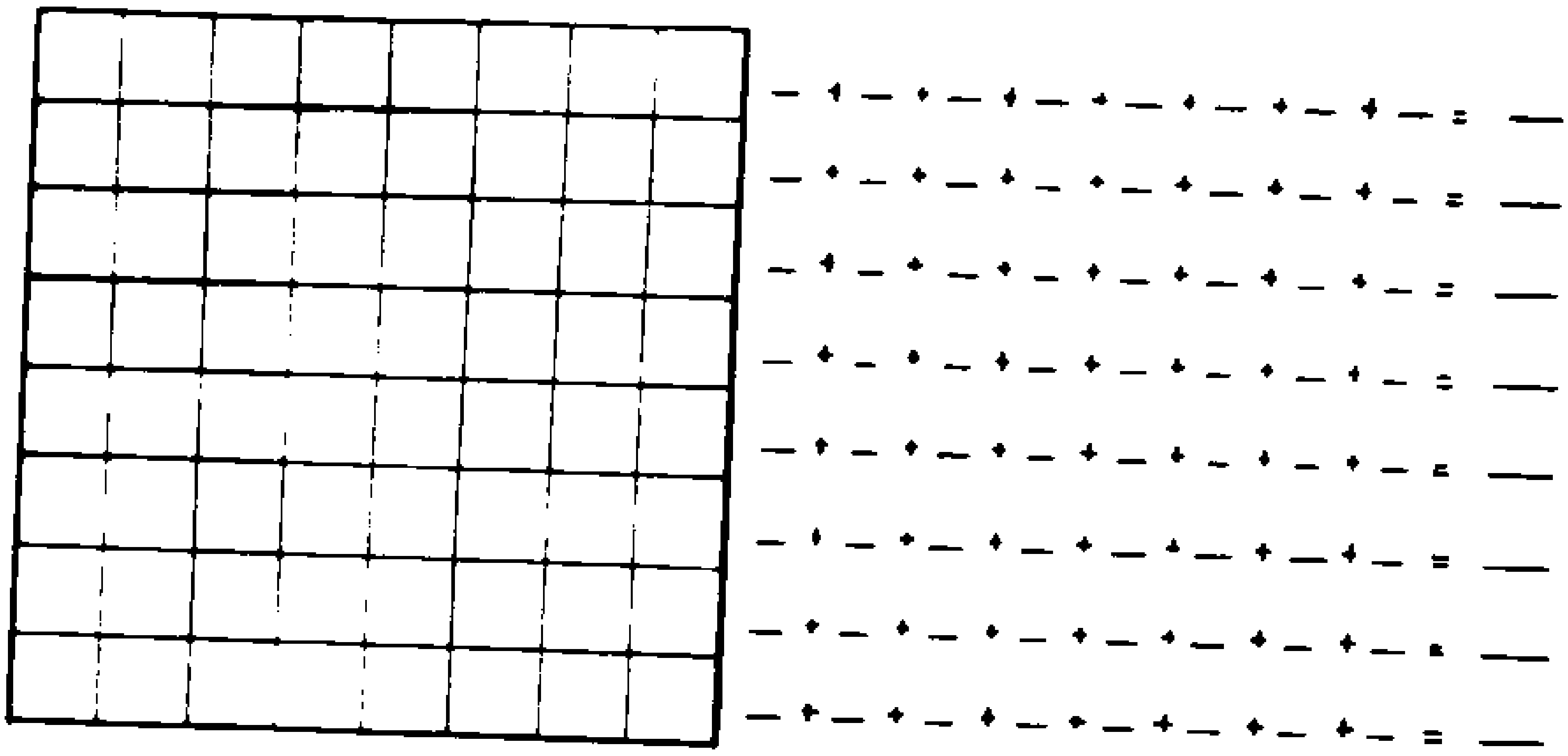
Het Japanse teken SI

Hoe is dit teken om te zetten in gegevens die de computer uit de lijst kan halen?

Allereerst wordt het teken uitvergroot getekend op een rooster van 8x8 hokjes. Elk hokje wordt zwart of wit. Vereenvoudiging van het teken is daarbij vaak noodzakelijk. Zie de tekening hierboven.

Het rooster moet in 8 horizontale rijen opgedeeld worden. Elke rij gaat een geheugenplaats bezetten. Acht geheugenplaatsen samen vormen de gegevens voor een letterteken. Elke rij wordt bepaald door een getal. De acht getallen geven aan welke hokjes van de rijen op het scherm moeten oplichten en welke niet.

In elke rij moeten de waarden van de in te kleuren hokjes opgeteld worden. Elke kolom heeft een eigen waarde. De meest rechtse kolom heeft de waarde 1. De kolom links daarvan heeft de waarde 2^1=2. De derde kolom heeft de waarde 2^2=4. De vierde kolom heeft de waarde 2^3=8. Dit gaat zo verder tot de meest linkse kolom die de waarde 27=128 heeft.



128 64 32 16 8 4 2 1

Rooster van 8 rijen, met kolomwaarden

Een hele rij zwart op de tekening (wit op het scherm) is dus:

128+64+32+16+8+4+2+1=255

Dat is niet toevallig precies het maximale getal dat in een byte (een geheugenplaats) past. *Zie ook bijlage 9.* Een hele rij wit op de tekening (en dus zwart of niet getekend op het scherm) is:

0+0+0+0+0+0+0+0=0

Achter de tekening van de SI vindt u de getallen die nodig zijn om dat teken samen te stellen.

Laten we eens proberen de gegevens van de SI in de lijst te plaatsen:

```
10 BEGINLIJST=PEEK(106)*256
20 POKE BEGINLIJST+9,0
30 POKE BEGINLIJST+10,97
40 POKE BEGINLIJST+11,1
50 POKE BEGINLIJST+12,98
60 POKE BEGINLIJST+13,2
70 POKE BEGINLIJST+14,4
80 POKE BEGINLIJST+15,24
90 POKE BEGINLIJST+16,96
100 PRINT :PRINT CHR$(33)
```

Dit programma gaat er van uit dat u het vorige programma gerund heeft en niet van scherm gewisseld hebt. Is dit wel het geval, dan moet u eerst de wijzer naar de nieuwe lettertekenlijst weer instellen door:

POKE 756,BEGINLIJST/256

In het programma ziet u dat er acht verschillende geheugenplaatsen van een waarde worden voorzien. Merk op dat er niet gekozen is voor de eerste acht geheugenplaatsen, maar voor het tweede stel van acht. De eerste acht geheugenplaatsen vormen de gegevens voor een spatie. Het gehele scherm staat vol met spaties. Als u daarvan de gegevens verandert, wordt bijna het gehele scherm gevuld met tekentjes. Verander in het programma de getallen 9 tot en met 16 in de regels 20 tot en met 90 door de getallen 1 tot en met 8, en kijk wat er gebeurt.
Het effect van dit programma wordt getoond, doordat de computer het nieuwe symbool door middel van een PRINT CHR$ opdracht op het scherm plaatst. U kunt het symbool ook op het scherm krijgen door het indrukken van een toets. Druk tegelijk de SHIFT toets en de 1 in. Normaal gesproken zou u nu het uitroepteken op het scherm krijgen. Doordat in de nieuwe lijst de gegevens voor het uitroepteken vervangen zijn door de gegevens van de SI, ziet u nu een SI op uw scherm verschijnen.

De lijst in de war

Zoals u al zag bij het eerste programma over het kopieren van de lijst, heeft de computer de gegevens van de lettertekens in een andere volgorde in het geheugen, dan de ASCII-codes aangeven. U moet hier op letten om te zorgen dat u niet het verkeerde teken van nieuwe gegevens voorziet. Het is immers het handigst om de gewone letters intakt te laten, zodat u die nog kunt gebruiken. De minder gebruikelijke tekens kunt u dan herdefinieren. De volgorde in het geheugen is als volgt:

Volgorde ASCII code	Volgorde in geheugen	Soort tekens
32- 95 0- 31 96-127	0- 63 64- 95 96-127	Hoofdletters, getallen leestekens Grafische tekens kleine letters, grafische tekens

Dit zorgt voor extra moeilijkheden bij het bepalen welke geheugenplaats welk gegeven moet krijgen om het juiste teken te veranderen. Het vermenigvuldigen met 8, vanwege het feit dat elk teken 8 bytes aan gegevens nodig heeft, was al duidelijk. Om het beginadres van elk teken te vinden kunt u de volgende formules gebruiken:

ASCII code (AC)	Geheugenlokatie (eerste van de acht lokaties)
32- 95 0- 31 96-127	BEGINLIJST + (AC-32)*8 BEGINLIJST + (AC+64)*8 BEGINLIJST + AC*8

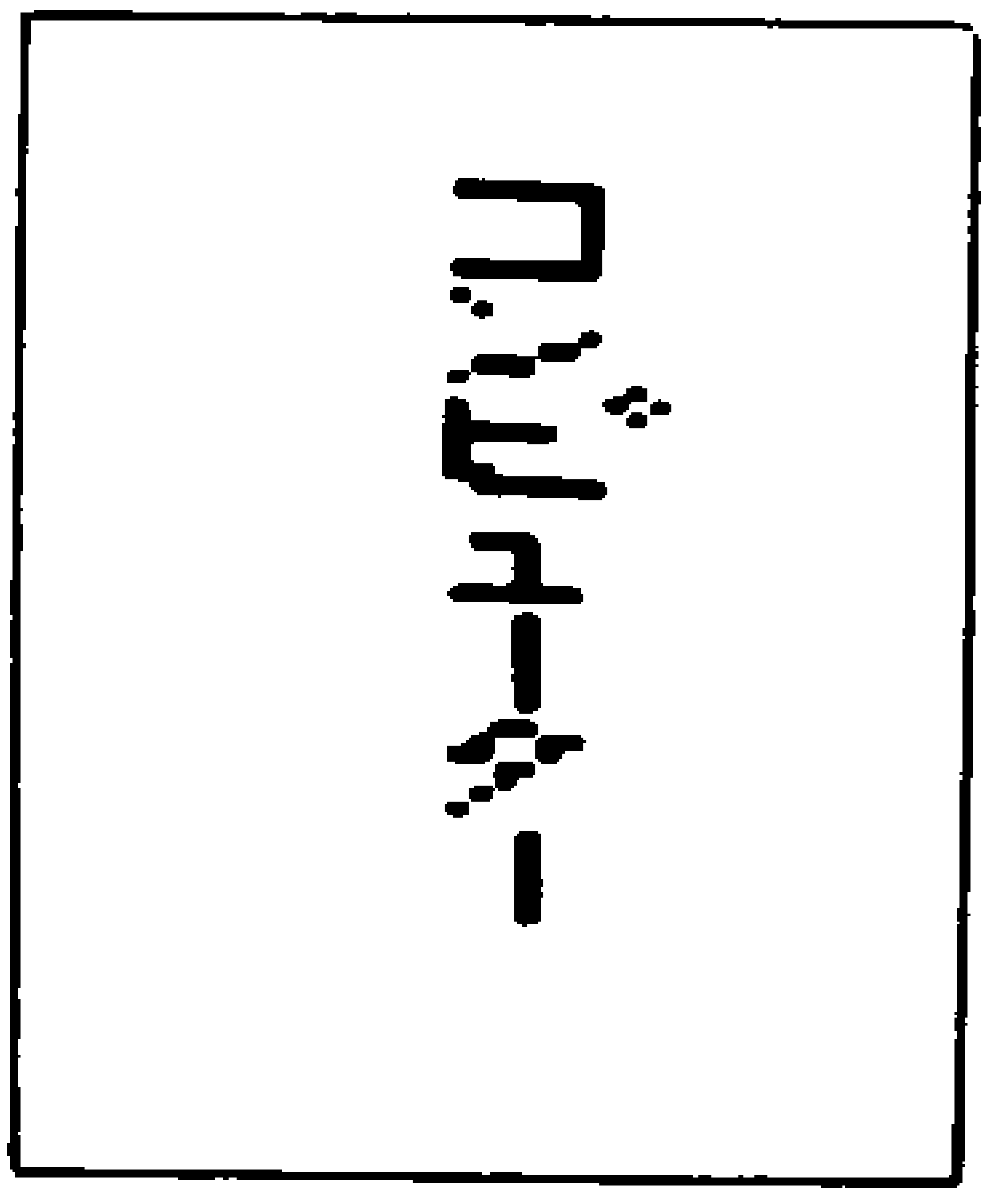
Bijvoorbeeld:
U wilt in het vervolg als u een '&' intikt, de griekse letter PI op het scherm zien. '&' heeft ASCII code 38. De gegevens van de PI moeten dus beginnen op geheugenlokatie **BEGINLIJST+(38-32)*8.**
Zie het onderstaande programma:

```
10 POKE 756,PEEK(106)
20 FOR A=0 TO 7
30 READ GE
40 POKE BEGINLIJST+48+A,GE
50 NEXT A
60 PRINT :PRINT CHR$(38)
70 POKE BEGINLIJST+14,4
80 POKE BEGINLIJST+15,24
90 POKE BEGINLIJST+16,96
100 DATA 0,0,1,102,84,20,20,20
```

Regel 10 begint met het verplaatsen van de wijzer, voor het geval u ondertussen van scherm gewisseld bent. Heeft u sinds het kopieren van de lijst met lettertekengegevens, de computer uit gehad, dan moet u eerst dat programma weer runnen.

Het volgende programma laat zien hoe alle lettertekens die samen het Japanse woord computer vormen, worden geladen en vervolgens op het scherm worden gezet.

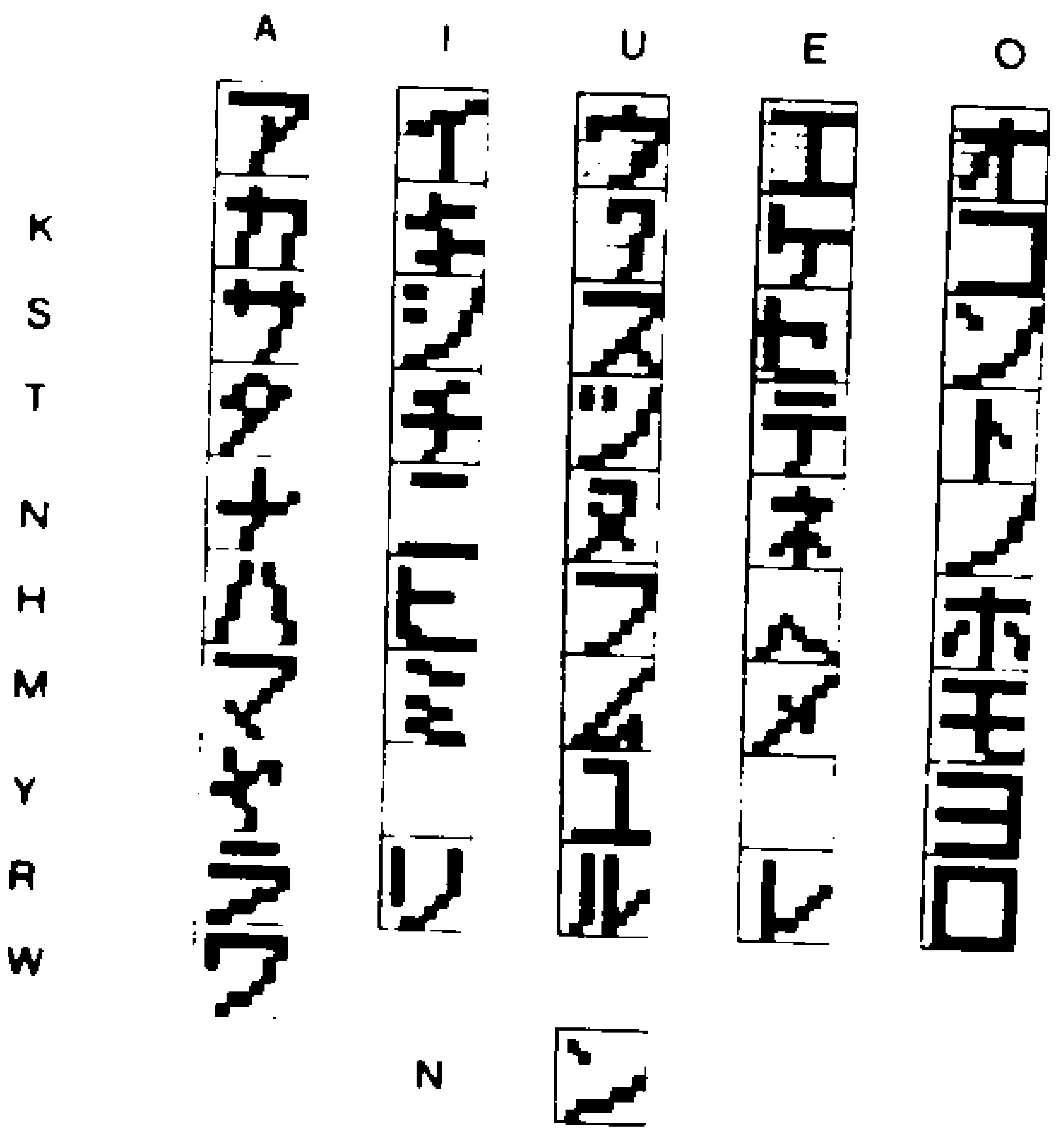
```
10 POKE 106,PEEK(106)-4:GRAPHICS 0
30 BEGINLIJST=PEEK(106)*256
40 POKE 756,BEGINLIJST/256
50 FOR KOPIE=0 TO 1023
60 POKE BEGINLIJST+KOPIE,PEEK(57344+KOPIE)
70 NEXT KOPIE
80 REM
90 FOR Q=33 TO 39
100 FOR A=0 TO 7
110 READ GE
120 POKE BEGINLIJST+((Q-32)*8)+A,GE
130 NEXT A
140 NEXT Q
150 PRINT :PRINT
160 PRINT "      ";CHR$(33)
170 PRINT "      ";CHR$(37)
180 PRINT "      ";CHR$(35);CHR$(39)
190 PRINT "      ";CHR$(36)
200 PRINT "      ";CHR$(38)
210 PRINT "      ";CHR$(34)
220 PRINT "      ";CHR$(38)
230 PRINT
500 DATA 0,127,1,1,1,1,1,127
510 DATA 0,24,38,100,24,16,32,64
520 DATA 0,64,64,124,64,64,96,63
530 DATA 0,0,0,56,8,8,8,126
540 DATA 0,64,32,0,1,6,56,64
550 DATA 0,8,8,8,8,8,8,8
560 DATA 64,160,64,0,0,0,0,0
```

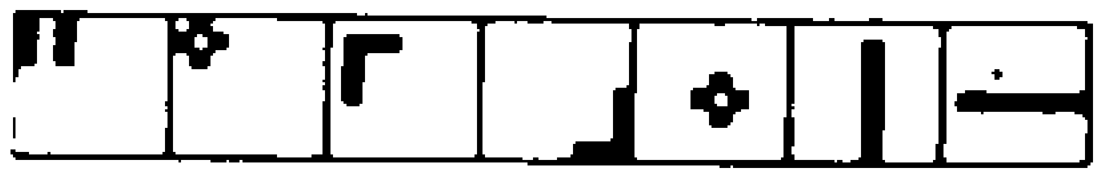


Door dit programma worden verschillende tekens veranderd, en vervolgens gebruikt om van boven naar beneden het japanse woord voor computer op het scherm te zetten. Als u het programma na het runnen nog een keer LIST, ziet u dat ook in de listing de leestekens zijn veranderd.
De gebruikte tekens zijn:

ASCII code	gemaakt teken
33	KO
34	TA
35	HI
36	YU
37	N
38	verlengstreep
39	Han-nigori (klankteken)

Computer wordt in het japans gevormd door een KO, gevolgd door een N, gevolgd door een PI met een han-nigori die de klank verbindt met het volgende teken. Dit is een YU, die gerekt wordt door een verlengstreep. Het totaal vormt dan PYUU. Tenslotte wordt de TA ook gerekt.
Voor diegenen onder u die andere woorden dan het voorbeeld in het japanse schrift willen zetten of die het schrift willen gebruiken als geheimschrift of als illustratie volgt hier de volledige tekenset zoals die in een computer geplaatst kan worden.

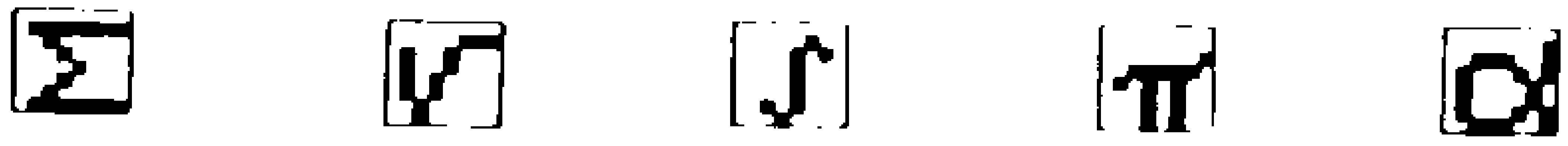




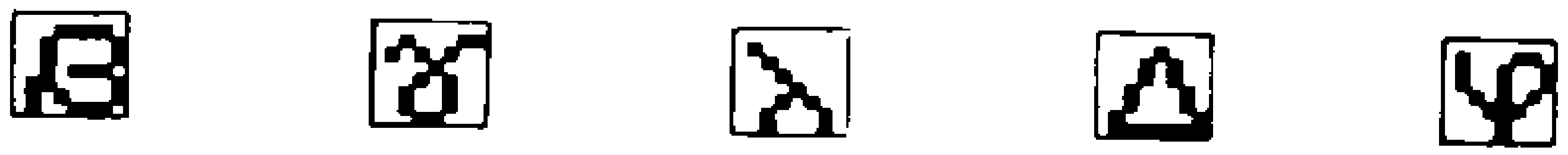
nigori han-nigori aanhalingstekens punt verlengstrepen

De japanse Katakana-set

Voor de meer wiskundige toepassingen hieronder een paar tekens die veel gebruikt worden bij wiskundige toepassingen:



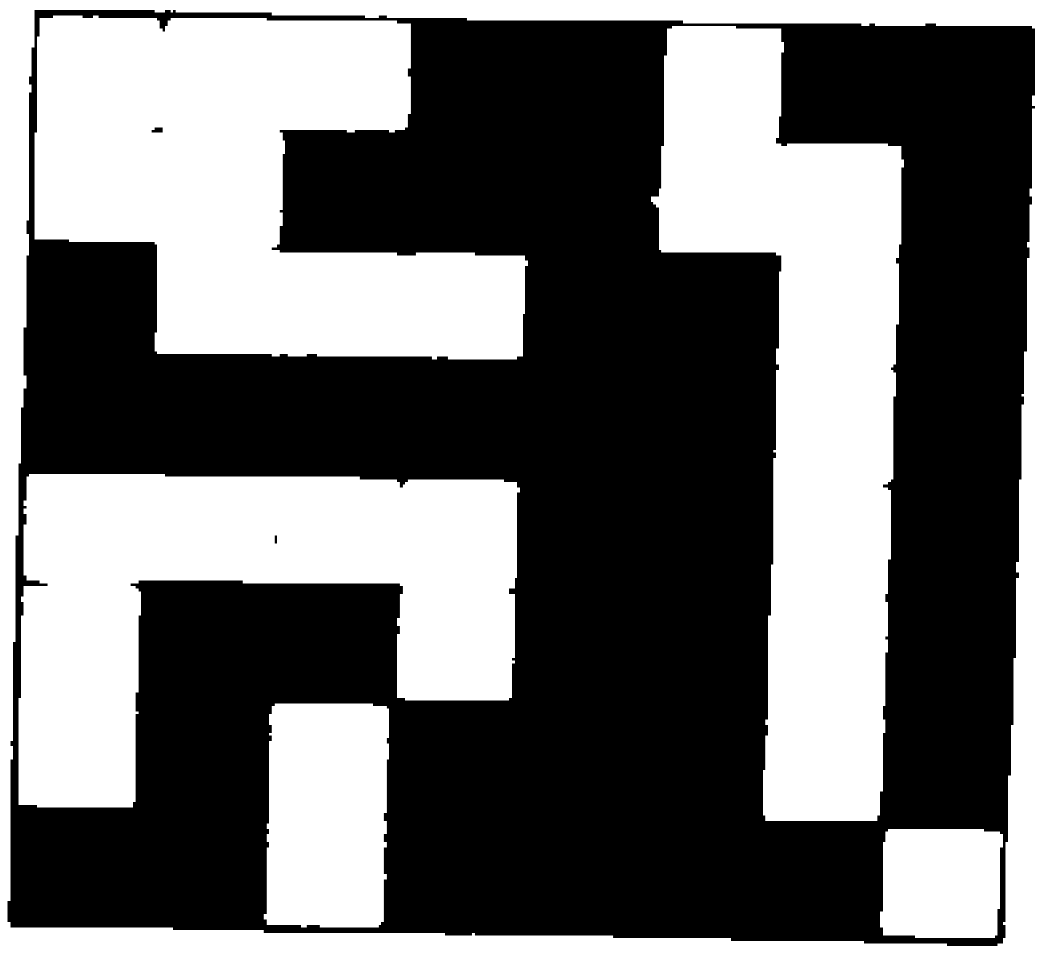
Sigma worteteken integraal pi alpha



beta gamma labda delta fi

Een aap

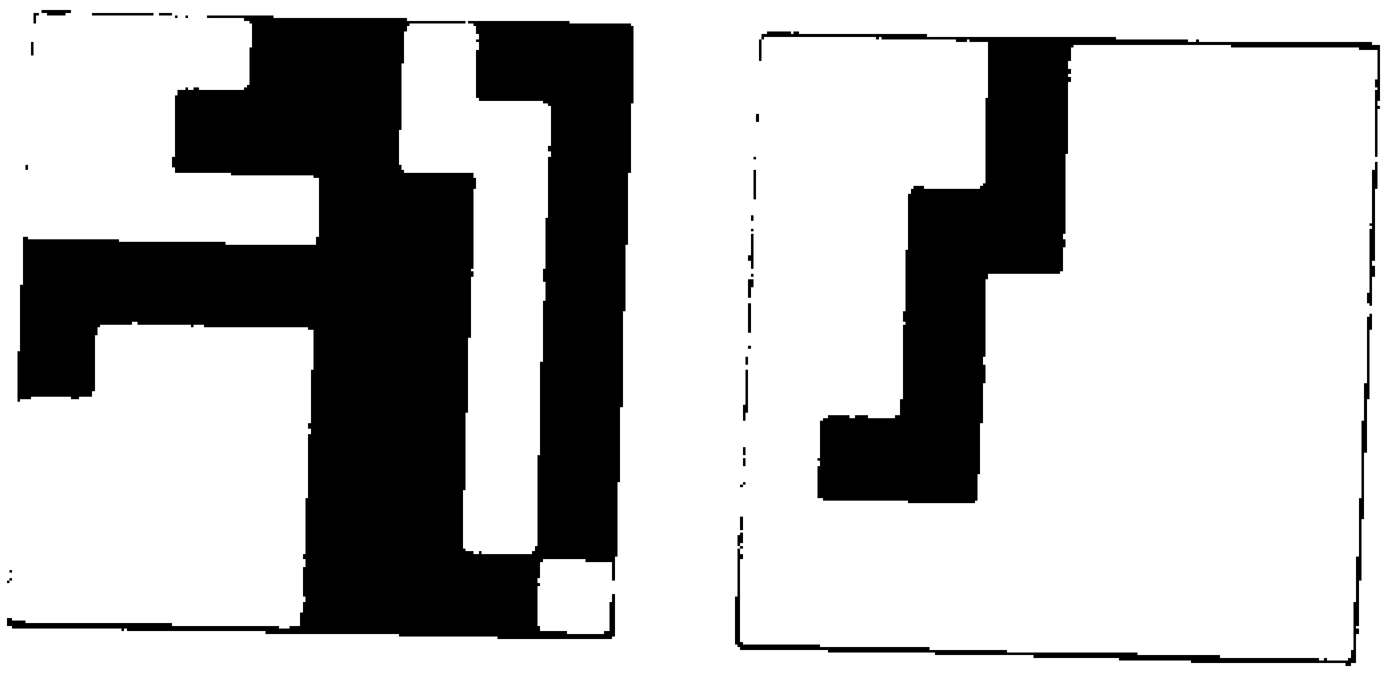
Naast deze min of meer serieuze toepassingen van het gebruik van zelf-maak tekens kunt u deze mogelijkheid van de Atari ook goed gebruiken voor allerlei grafische effecten. Als voorbeeld dit aapje:



Een aapje

De codes die nodig zijn voor dit aapje: 27,57,141,253,13,109,93,222

Nu beschouwt de computer dit aapje als een gewoon letterteken. Dat wil zeggen dat we het op een willekeurige plaats op het scherm kunnen zetten door een POSITION-opdracht gevolgd door een PRINT CHR$-opdracht.
Om te laten zien dat hiermee leuke animatie mogelijk is, moet u de volgende twee tekeningen ook als letterteken definieren.



Extra aap-onderdelen

De codes zijn: 27,57,13,253,141,13,13,14
16,16,48,32,32,96,0,0

We kunnen dit alles samenvoegen tot een programma dat de aap laat bewegen.

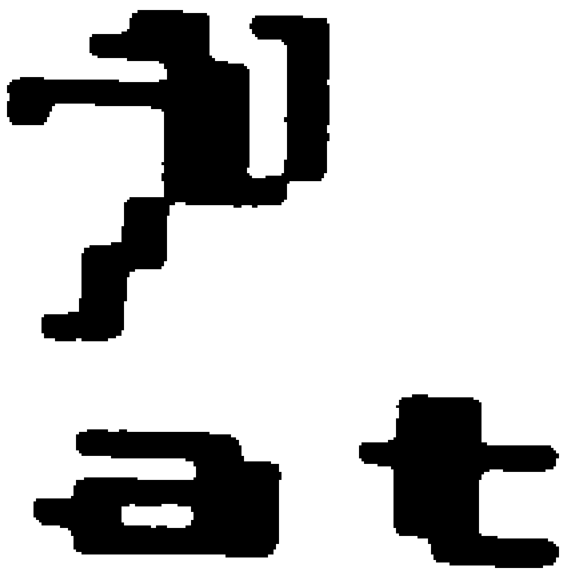
```
10 POKE 106,PEEK(106)-4:GRAPHICS 0
20 FOR A=1 TO 124:PRINT CHR$(A);:NEXT A
30 BEGINLIJST=PEEK(106)*256
40 POKE 756,BEGINLIJST/256
50 FOR KOPIE=0 TO 1023
60 POKE BEGINLIJST+KOPIE,PEEK(57344+KOPIE)
70 NEXT KOPIE
80 GRAPHICS 0:POKE 752,1
90 POKE 756,BEGINLIJST/256
100 FOR Q=1 TO 3
110 FOR A=0 TO 7
120 READ GE
130 POKE BEGINLIJST+(Q+64)*8+A,GE
140 NEXT A
150 NEXT Q
160 REM TEKENEN
170 G=22
180 FOR H=19 TO 7 STEP -2
190 POSITION H,G:PRINT CHR$(1)
200 FOR WA=1 TO 120:NEXT WA
210 POSITION H,G:PRINT CHR$(3)
220 POSITION H,G-1:PRINT CHR$(2)
230 FOR WA=1 TO 40:NEXT WA
240 POSITION H,G:PRINT " "
250 POSITION H,G-1:PRINT " "
```

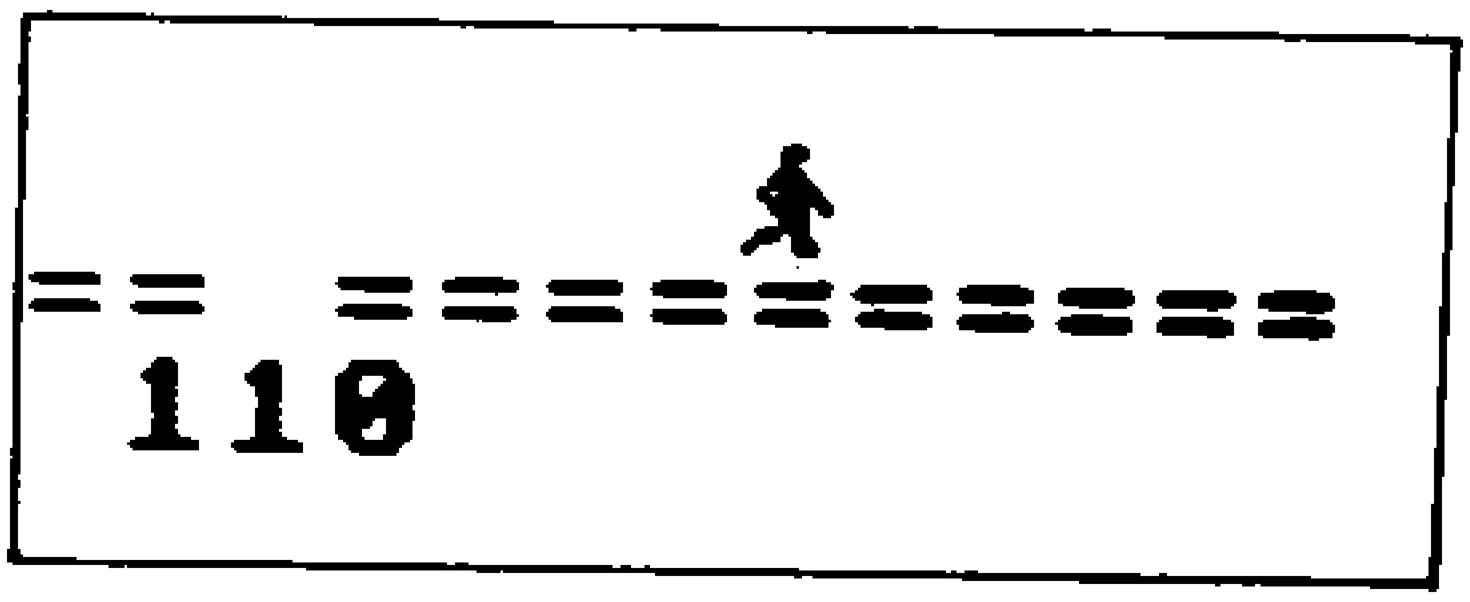
```
260 POSITION H-1,G-1:PRINT CHR$(3)
270 POSITION H-1,G-2:PRINT CHR$(2)
280 FOR WA=1 TO 40:NEXT WA
290 POSITION H-1,G-1:PRINT " "
300 POSITION H-1,G-2:PRINT " "
310 LET G=G-3
320 NEXT H
500 DATA 27,57,141,253,13,109,93,222
510 DATA 27,57,13,253,141,13,13,14
520 DATA 16,16,48,32,32,96,0,0
```

Het eerste deel van het programma (tot en met regel 70) kan weg worden gelaten als u de lijst met gegevens voor de lettertekens al gekopieerd heeft, en de computer ondertussen niet uit gehad heeft.
Regel 80 zorgt voor het wissen van het scherm en het uitzetten van de cursor.
Regel 90 stelt de wijzer opnieuw in op de kopie van de lettertekenlijst. Dit moet telkens na het wissen van het scherm gebeuren!
De regels 100-150 zorgen voor het invoeren van nieuwe lettertekens in de vorm van een aapje.
De regels 160-320 zorgen voor het telkens opnieuw tekenen van het aapje en vervolgens weer voor het uitwissen.
De laatste drie regels bevatten de gegevens die nodig zijn om de verschillende onderdelen van het aapje te tekenen.

De woeste rivier

Met de hier gegeven technieken kan een leuk spelletje gemaakt worden. U bent op de vlucht, of u moet een prinses in nood helpen. Kortom u moet een gevaarlijke rivier vol ijsschotsen oversteken. U moet van schots naar schots springen. De spatiebalk zorgt ervoor dat u omhoog gaat. Het is het veiligst om de spatiebalk zo vaak mogelijk in te drukken, maar u krijgt alleen punten voor het aantal stappen dat u op een schots doet.
De lezer kan eventueel een tijdslimiet inbouwen. De subroutine die begint bij 1000 zorgt ervoor dat de rivier er telkens anders uitziet. De subroutine bij 2000 kijkt of u niet in het water valt. En pas op! U weet hoe dat met ijs gaat: op het randje staan is heel gevaarlijk!





```
   POKE 1[illegible]6,PEEK(106)-4:GRAPHICS 0
2[illegible] FOR A=1 TO 124:PRINT CHR$(A);:NEXT A
3[illegible] BEGINLIJST=PEEK(106)*256
4[illegible] POKE [illegible]56,BEGINLIJST [illegible]256
5[illegible] FOR KOPIE=[illegible]
6[illegible] POKE BEGIN[illegible]+KOPIE,PEEK(573[illegible]4+KOPIE)
7[illegible] NEXT KOPIE
[illegible] OPEN #2,4,[illegible]
9[illegible] GRAPHICS [illegible] POKE [illegible]
[illegible] POKE [illegible]56,B[illegible]NLIJST [illegible]256
[illegible] RESTORE
[illegible] FOR [illegible]
[illegible] FOR A=[illegible]
[illegible] READ GE
[illegible] POKE BEGINLIJST+[illegible]*8+A,GE
[illegible] NEXT A
[illegible] NEXT [illegible]
16[illegible] REM SPEL
1[illegible] SCORE=[illegible]
1[illegible] GOSUB [illegible]
[illegible] POSITION [illegible],8[illegible]PRINT CHR$(1)
[illegible] FOR Z=1 T[illegible]:NEXT [illegible]
2[illegible] POSITION [illegible]:PRINT CHR$(2)
22[illegible] FOR Z=1 T[illegible]:NEXT Z
230 POSITION LR,BO:PRINT " "
2[illegible] BO=[illegible]:IF PEEK([illegible])=255 THEN 260
2[illegible] GET #2,SPAT:IF SPAT=32 THEN BO=13
260 LR=LR+1
2[illegible] IF LR=39 THEN GOSUB [illegible]
2[illegible] IF BO=14 THEN S[illegible]RE=SCORE+1
2[illegible] POSITION [illegible]:PRINT SCORE
300 GAT=0
3[illegible] GOSUB 2[illegible]
3[illegible] IF GAT [illegible] THEN [illegible]
33[illegible] GOSUB 3[illegible]
3[illegible] FOR WA=1 T[illegible]:NEXT WA
35[illegible] GOTO [illegible]
36[illegible] END
3[illegible] REM SUBR[illegible]TINES
1000 LR=0:BO=[illegible]=INT(RND(0)*4)+1
1[illegible] IF [illegible]=1 THEN POSITION 0,15:? "▬▬▬▬▬▬▬▬▬  ▬▬▬▬▬ ▬▬▬▬▬▬▬▬▬▬  ▬▬▬▬▬▬▬▬▬▬"
1[illegible]2[illegible] IF [illegible]=2 THEN POSITION 0,15:? "▬▬▬▬ ▬▬▬▬▬▬▬▬▬▬▬  ▬▬▬ ▬▬▬▬▬▬▬ ▬▬▬▬▬▬▬▬▬"
1[illegible]3[illegible] IF [illegible]=3 THEN POSITION 0,15:? "▬▬▬▬▬  ▬▬▬▬ ▬▬ ▬▬▬▬▬ ▬▬▬▬▬▬▬▬▬▬ ▬▬▬▬▬▬▬"
1[illegible]4[illegible] IF [illegible]=4 THEN POSITION 0,15:? "▬▬▬▬▬▬▬▬▬▬▬▬▬    ▬▬▬▬▬▬▬ ▬▬▬▬ ▬▬▬▬▬▬ ▬▬▬"
1050 RETURN
1[illegible]60 REM
2[illegible] IF [illegible]=1 AND BO=14 AND (LR=9 OR LR=10 OR LR=16 OR LR=27 OR LR=28) THEN GAT=1
2010 IF [illegible]=2 AND BO=14 AND (LR=4 OR LR=16 OR LR=17 OR LR=21 OR LR=29) THEN GAT=1
2[illegible]2[illegible] IF [illegible]=3 AND BO=14 AND (LR=5 OR LR=6 OR LR=14 OR LR=20 OR LR=31) THEN GAT=1
2030 IF [illegible]=4 AND BO=14 AND (LR=13 OR LR=14 OR LR=15 OR LR=23 OR LR=28 OR LR=35) THEN GAT=1
2[illegible] RETURN
2[illegible]50 REM
3000 FOR BO=1[illegible] T[illegible]
3010 PRINT CHR$(12[illegible]);CHR$(253);
3020 POSITION LR,BO:PRINT CHR$(1)
3030 FOR WA=1 TO [illegible]:NEXT WA
3040 POSITION LR,BO:PRINT " "
3050 NEXT BO
3060 RETURN
4000 DATA 24,1[illegible],56,84,58,24,104,1[illegible]
4010 DATA 24,1[illegible],56,[illegible],24,20,24,24
```

16. GRAFIEK EN KLEUR

Inleiding

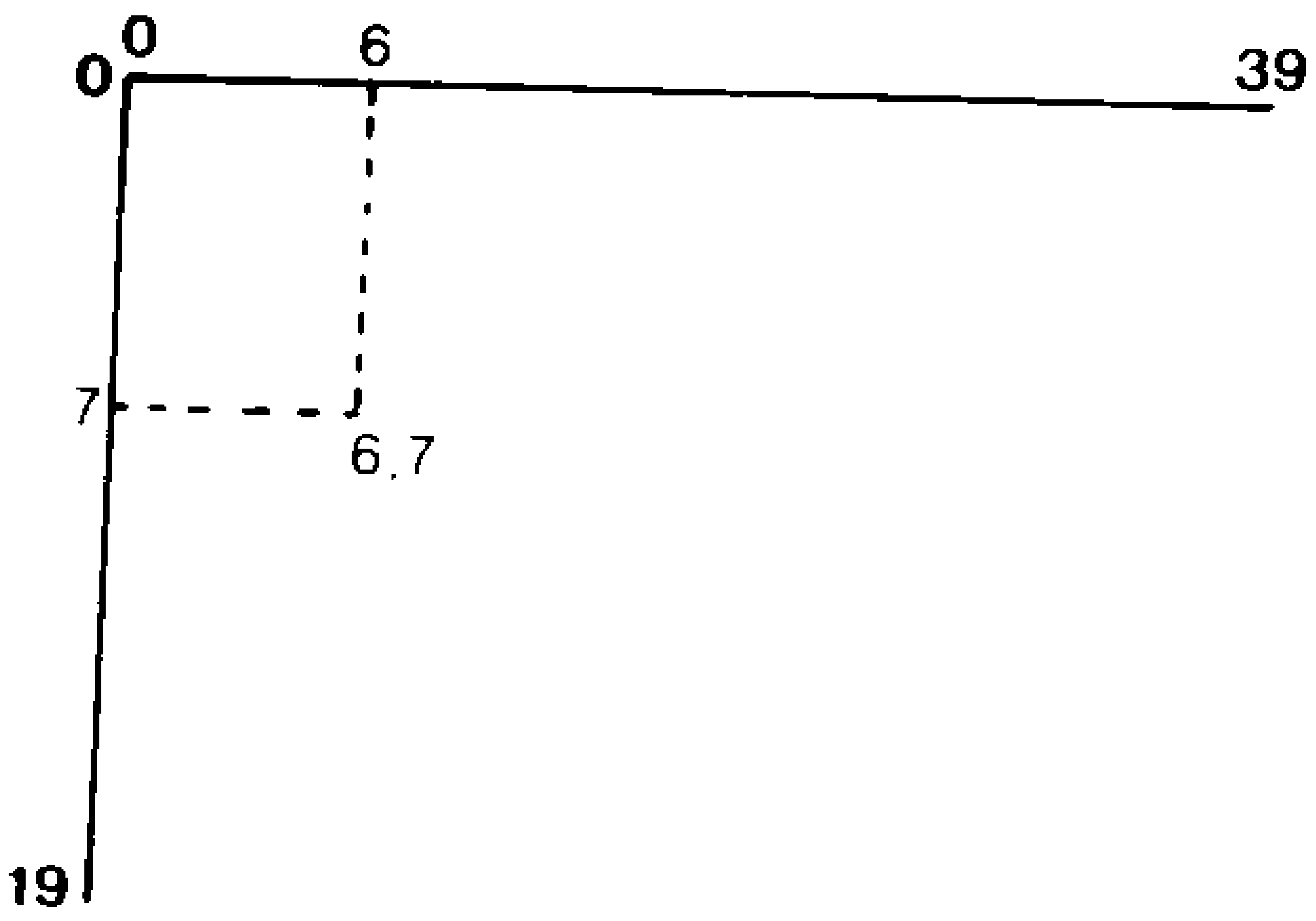
In hoofdstuk 14 las u al dat schermmodus 3 geschikt is voor het weergeven van kleuren. Daarnaast heeft scherm 3 het 'voordeel' dat het alleen blokjes kent en geen stippen. Een lijn op scherm 3 getrokken ziet er dan ook uit als een serie tamelijk grove blokken. Enerzijds is dit een nadeel omdat op deze wijze in scherm drie geen fijne lijnen getrokken kunnen worden, maar anderzijds kunt u er leuke effecten mee bereiken.
Voor het tekenen van een mannetje hoeft u alleen een stel lijnen in de juiste kleur te trekken. De computer vult de blokken in waardoor het mannetje vaste vorm krijgt. Probeert u het volgende programma maar eens.

```
10 GRAPHICS 3
20 COLOR 2
30 PLOT 12,1
40 FOR A=1 TO 18
50 READ X,Y,Z
60 IF Z>0 THEN DRAWTO X/4,Y/2:GOTO 70
65 PLOT X/4,Y/2
70 NEXT A
500 DATA 44,9,1,44,2,1,44,9,0,36,23,1,28
505 DATA 27,1,10,35,1,10,38,1,36,23,0
510 DATA 52,22,1,60,31,1,66,31,1,44,9,0
515 DATA 26,13,1,26,17,1,44,9,0,50,17,1
520 DATA 60,17,1,64,15,1
```

De centrale opdracht bij het maken van een tekening is

DRAWTO x,y

Deze opdracht zorgt ervoor dat de computer een lijn van het ene punt naar het andere punt trekt. De punten worden opgegeven in een rechthoekig coordinatenstelsel. Van links naar rechts loopt de X-coordinaat van 0 tot en met 40. Van boven naar onder loopt de Y-coordinaat van 0 tot en met 20. Elke schermmodus heeft zijn eigen coordinatenstelsel. De maximale waarden van de X-coordinaat en de Y-coordinaat verschillen telkens. Ze beginnen echter altijd allebei bij 0, en allebei in de linkerbovenhoek. Voor de verschillen tussen de schermmodi wordt u verwezen naar hoofdstuk 14. Voor een schermmodus 3 zou het assenstelsel er zo uit zien:



In dit assenstelsel ziet u aangegeven hoe u een bepaald punt door middel van twee getallen kunt aangeven. Het eerste getal is altijd de X-coordinaat (van links naar rechts) en het tweede getal is altijd de Y-coordinaat (van boven naar beneden). De lezers die wiskunde gehad hebben worden extra gewaarschuwd: de schaal van de Y-AS loopt VAN BOVEN NAAR BENEDEN!
De DRAWTO opdracht heeft twee getallen nodig. Het eerste getal geeft de X-coordinaat van het punt waar de lijn naar toe getrokken moet worden aan. Het tweede getal geeft de Y-coordinaat van dit punt aan. De lijn wordt getrokken vanaf de plaats van de grafische cursor op het moment dat de opdracht uitgevoerd gaat worden.
De grafische cursor is vergelijkbaar met de tekstcursor. Het is een plaatsbepaler die bepaalt waar het eerst volgende lichtpuntje ingekleurd gaat worden. De grafische cursor is echter niet zichtbaar.
De lijn wordt getrokken vanaf de grafische cursor naar het opgegeven punt. Na de opdracht staat de grafische cursor op het laatst getekende punt. De computer heeft een lijn naar dat punt getrokken, en de grafische cursor is meegegaan. Soms is het echter handiger om een lijn niet te trekken vanaf de huidige plaats van de grafische cursor, maar vanaf een ander punt. Om te zorgen dat de grafische cursor verplaatst kan worden, dient de opdracht:

PLOT x,y

Het eerste getal geeft weer de X-coordinaat van een punt aan, en het tweede getal de Y-coordinaat van dat punt. Op het aangegeven punt tekent de computer een punt in de kleur die opgegeven is door de opdracht

COLOR kl

De letters 'kl' staan voor het opgeven van een kleur. Deze waarde verwijst naar een kleurregister en kan bij de normale schermmodi een waarde hebben van 0 tot en met 4. In de schermmodi 9, 10 en 11 zijn grotere waarden mogelijk. *Zie hoofdstuk 14.*
Het volgende programma demonstreert dat de werking van een assenstelsel bij de verschillende schermen gelijk is. Tik het in en aanschouw het effect.

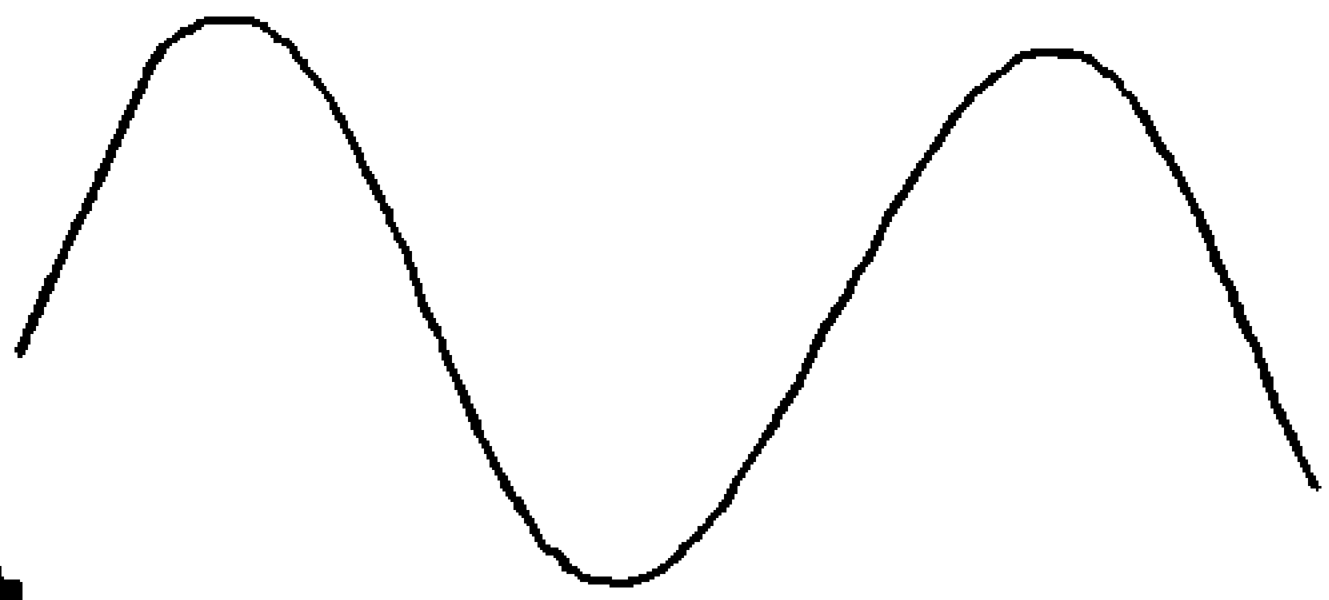
```
10 GRAPHICS 3
20 FOR A=1 TO 300
30 X=INT(RND(0)*40)
40 Y=INT(RND(0)*20)
50 KL=INT(RND(0)*4)
60 COLOR KL
70 PLOT X,Y
80 NEXT A
90 GOTO 90
```

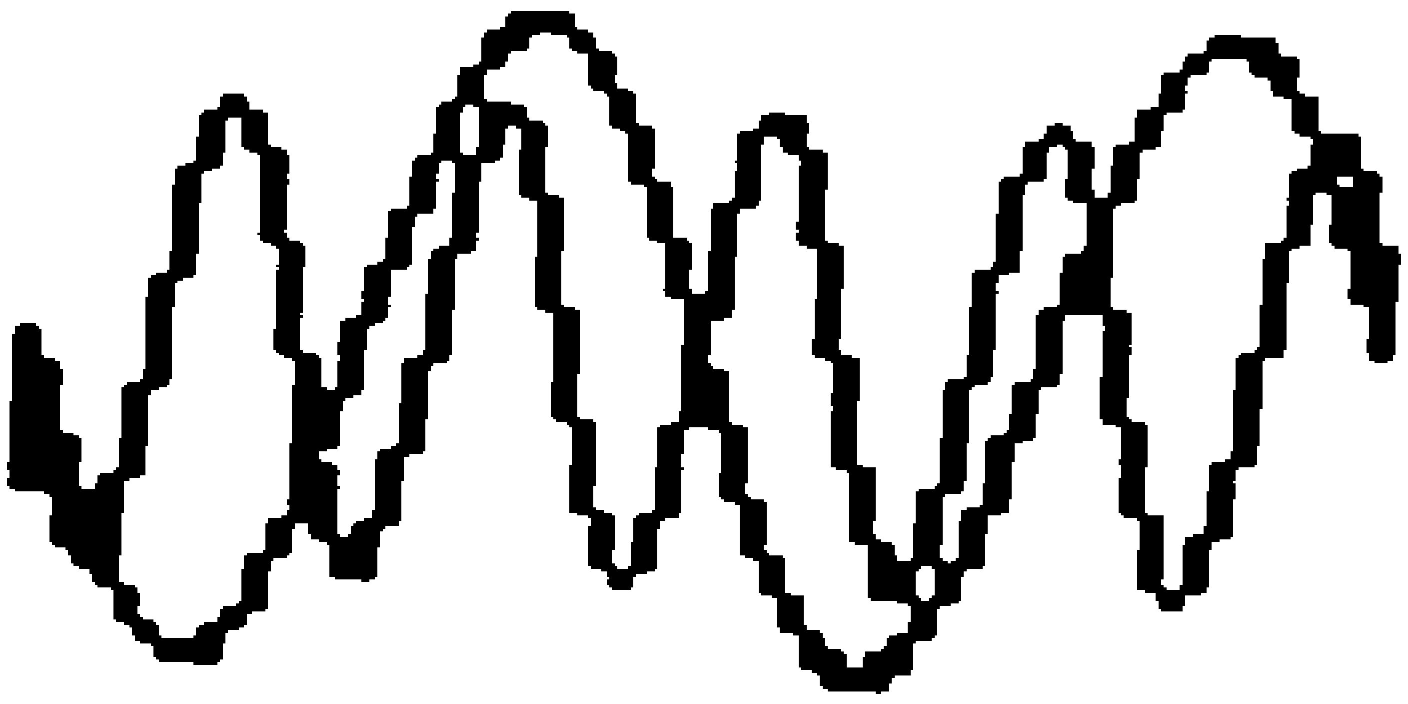
Dit programma zet driehonderd keer een punt. De punt wordt telkens op een willekeurige plaats gezet. De regels 30 en 40 zorgen voor een willekeurige X-coordinaat en een willekeurige Y-coordinaat. Regel 50 zorgt voor een willekeurige kleur.
Als u het programma runt zoals het hier staat, krijgt u een heleboel hele gekleurde blokjes te zien. Dat zijn de lichtpunten van scherm 3, in kleur, maar met een gering oplossend vermogen.
U kunt het programma eenvoudig aanpassen om de stippen van de andere schermen te zien. Probeer eerst in regel 10 de GRAPHICS 3 te veranderen in GRAPHICS 5 en later in GRAPHICS 7. De kleuren van de blokjes blijven hetzelfde, maar de stippen worden steeds kleiner. Dit wordt veroorzaakt doordat het raster van deze schermmodi fijner is. Doordat zowel de X-as als de Y-as langer zijn bij deze schermmodi, gebruikt dit programma dan slechts een deel van het scherm. Pas dit aan door middel van de twee vermenigvuldigingsfactoren in de regels 30 en 40. Probeer dit voor alle schermmodi uit. U kunt nu fraai zien hoe een stip er op elk scherm uitziet. Let ook op het speciale effect dat scherm 12 en 13 veroorzaken.
Het volgende programma laat zien dat ook in schermmodus 3 hele fraaie en soms ook nuttige grafieken gemaakt kunnen worden. Het volgende programma tekent een grafiek van twee sinusfunkties en een cosinusfunktie door elkaar heen. Doordat elke grafiek zijn eigen kleur heeft, zijn ze goed te onderscheiden.

```
10 GRAPHICS 3+16
20 COLOR 1
30 PLOT 0,11
40 FOR X=0 TO 12.5 STEP 0.1
50 DRAWTO X*3,SIN(X)*10+11
60 NEXT X
70 COLOR 2
80 PLOT 0,10
90 FOR X=0 TO 12.5 STEP 0.1
100 DRAWTO X*3,SIN(X*2.5)*6+11
110 NEXT X
120 COLOR 3
130 PLOT 0,10
140 FOR X=0 TO 12.5 STEP 0.1
150 DRAWTO X*3,COS(X)*10+11
```

```
160 NEXT X
170 GOTO 170
```

De grafieken worden getekend door de regels 30-60, 80-110 en 130-160. De PLOT opdrachten in het programma dienen om de grafische cursor telkens aan het begin van de grafiek te zetten.





Wilt u het programma versnellen dan kunt u in de regels 40, 90 en 140 de stapgrootte van de lus vergroten tot .2 of .3. Hoe groter de stap, hoe groter de onnauwkeurigheid van de grafiek. Maar bij .2 als stapgrootte is dat nog niet te zien.

Let u op de wijze waarop de normale sinusberekening is aangepast om te zorgen voor een mooie weergave van de grafiek. Bij een onbehandelde grafiek zou de Y-coordinaat altijd de waarde van SINus(X) hebben. De waarde van deze sinusfunktie ligt altijd tussen -1 en 1. Dat betekent een verschil van twee. Als we dit direkt op het scherm willen zetten, krijgen we bijna een rechte lijn. Daarom vergroten we de afstand tussen de minimum en maximum Y-waarde door een vergrotingsfactor. In bovenstaand programma werd dat: SIN(X)*10, SIN(X*2.5)*6 en COS(X)*10. De SIN(X) en COS(X) funktie werden evenveel vergroot en de SIN(X*2.5) werd iets minder vergroot. Op dezelfde wijze werd de grafiek in de X-richting vergroot. Voor alle drie de grafieken was dat X*3. Het totaalresultaat is dat de grafiek meer in de Y-richting dan in de X-richting vergroot is. De grafieken zijn daardoor een beetje vervormd.
Als laatste aanpassing is de hele grafiek een eind naar beneden verschoven. Dit is gedaan omdat zowel de sinus als de cosinusgrafiek ook negatieve uitkomsten hebben. Deze zouden dan bovenaan het scherm wegvallen. Let op! De Y-as loopt van boven naar beneden en niet van beneden naar boven. De grafieken staan onderste boven. Het verschuiven gebeurt door telkens 11 op te tellen bij de Y-coordinaat. Zie de regels 50, 100 en 150.

Het programma maakt duidelijk dat de kleuren van scherm 3 wel handig zijn voor het onderscheiden van verschillende grafieken, maar dat het laag oplos-

send vermogen evenzeer een belemmering voor goed afleesbare grafieken is. Voor het afbeelden van gekleurde grafieken kunt u beter gebruik maken van scherm 7:

```
10 GRAPHICS 7+16
20 COLOR 1
30 PLOT 0,41
40 FOR X=0 TO 12.5 STEP 0.1
50 DRAWTO X*12,SIN(X)*40+41
60 NEXT X
70 COLOR 2
80 PLOT 0,40
90 FOR X=0 TO 12.5 STEP 0.1
100 DRAWTO X*12,SIN(X*2.5)*24+41
110 NEXT X
120 COLOR 3
130 PLOT 0,79
140 FOR X=0 TO 12.5 STEP 0.1
150 DRAWTO X*12,COS(X)*40+41
160 NEXT X
170 GOTO 170
```

Taartdiagram

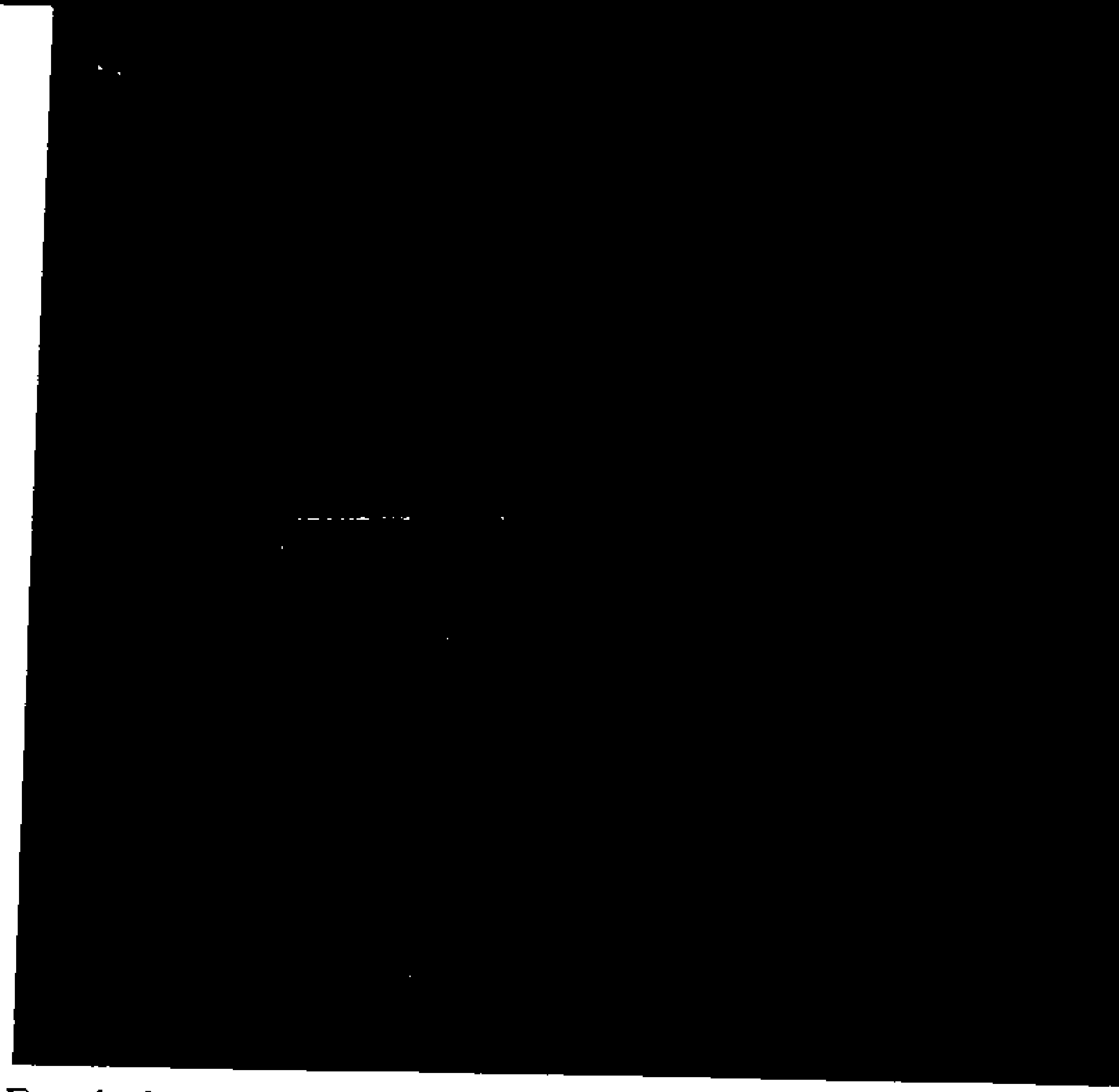
Het volgende programma toont u een meer serieuze toepassing van de kleuren van schermmodus 7.
Stel u heeft het hele jaar door koekjes, zuurtjes, ijs en drop gekocht. U heeft telkens na aankoop het bedrag dat met elk van de snoepsoorten gemoeid was, genoteerd. Aan het einde van het jaar wilt u zien hoe de onderlinge verhoudingen tussen de soorten bedragen ligt. Om onderlinge verhoudingen tussen verschillende grootheden grafisch aan te geven, is een zogenaamde 'taart' grafiek zeer geschikt. Deze grafiek geeft elke soort aan als een kleurtje. De oppervlakte van een bepaalde kleur geeft aan hoe groot het aandeel van een bepaalde soort in het totaal was. U ziet deze grafieken elk jaar na prinsjesdag in de krant die aangeven hoe groot deel van de totale rijksbegroting opgeslokt wordt door elk van de departementen.

```
10 GRAPHICS 7
20 A=2492:B=646:C=1893:D=423
30 SOM=A+B+C+D
40 STRAAL=35:PLAATS=0:STAP=6.28/(SOM/20)
50 COLOR 0:PLOT 80,40
60 COLOR 1
70 HOEK=6.28*(A/SOM)
80 FOR LU=PLAATS TO PLAATS+HOEK STEP STAP
90 DRAWTO STRAAL*COS(LU)+80,STRAAL*SIN(LU)+40
```

```
100 PLOT 80,40
110 NEXT LU
115 PLAATS=PLAATS+HOEK
120 COLOR 2
130 HOEK=6.28*(B/SOM)
140 FOR LU=PLAATS TO PLAATS+HOEK STEP STAP
150 DRAWTO STRAAL*COS(LU)+80,STRAAL*SIN(LU)+40
160 PLOT 80,40
170 NEXT LU
180 PLAATS=PLAATS+HOEK
190 COLOR 3
200 HOEK=6.28*(C/SOM)
210 FOR LU=PLAATS TO PLAATS+HOEK STEP STAP
220 DRAWTO STRAAL*COS(LU)+80,STRAAL*SIN(LU)+40
230 PLOT 80,40
240 NEXT LU
250 POSITION 0,0:PRINT "ROOD  = KOEKJES"
260 PRINT "GEEL  = ZUURTJES"
270 PRINT "BLAUW = IJS"
280 PRINT "ZWART = DROP";
290 GOTO 290
```

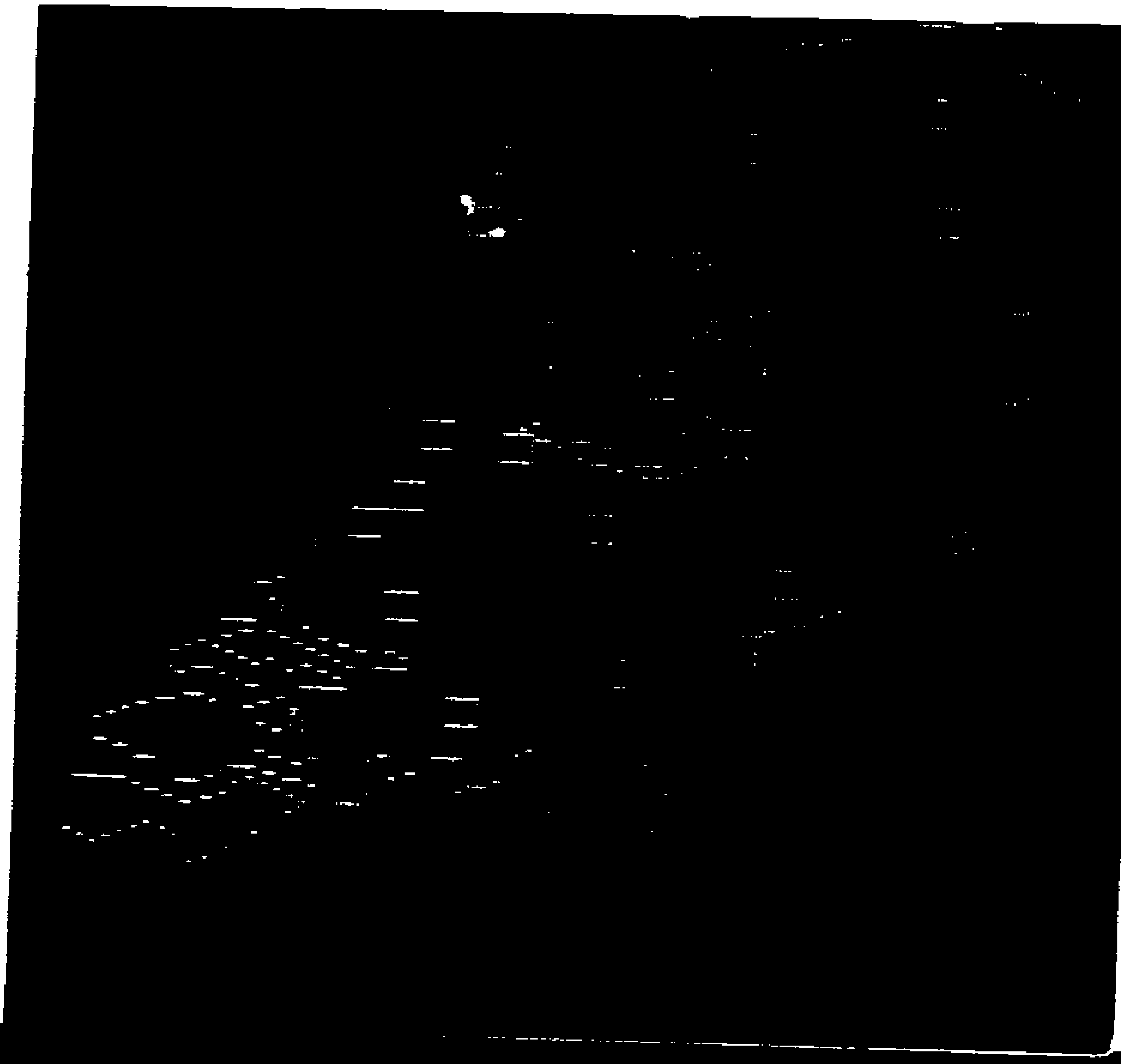
In regel 20 vindt u de uiteindelijke uitgaven voor de vier soorten snoep. laten we er maar van uitgaan dat het in centen uitgedrukt is. Natuurlijk kunt u hier ook elke andere soort grootheid vermelden, met eigen getallen. Regel 30 rekent het totaal uit. Elk van de grootheden moet immers als een gedeelte van het totaal aangegeven worden.
Regel 40 stelt de straal van de cirkel in. Daarna worden de beginplaats (0) en de stapgrootte ingesteld.
Regel 50 plaatst de grafische cursor in het midden van het scherm. De regels 70, 130 en 200 bepalen hoe groot deel van de cirkelomtrek bij een bepaalde grootheid hoort.
Regel 80 laat de grafische cursor langs de cirkelomtrek lopen en telkens een lijn trekken van het middelpunt van de cirkel naar de omtrek. Dit gaat net zolang door totdat er op de cirkelomtrek een boog beschreven is die groot genoeg is om het deel van de betreffende soort aan te geven.
Regel 90 tekent, regel 100 verplaatst de cursor weer naar het middelpunt. Regel 115 zorgt ervoor dat het volgende stuk cirkelboog genomen wordt.

De regels 120-180 en 190-240 doen hetzelfde als de regels ervoor, maar dan voor de andere taartstukken.
Het laatste stuk taart is in de achtergrondkleur, en hoeft dus niet getekend te worden. Wilt u meer dan vier soorten in de grafiek onderbrengen, dan zult u bepaalde kleuren meer dan een keer moeten gebruiken. Of u moet gebruik maken van scherm 10 of 11.
De regels 250-280 drukken de gegevens over de taart af, zodat de gebruiker kan zien welk kleurtje, naar welk gegeven verwijst.



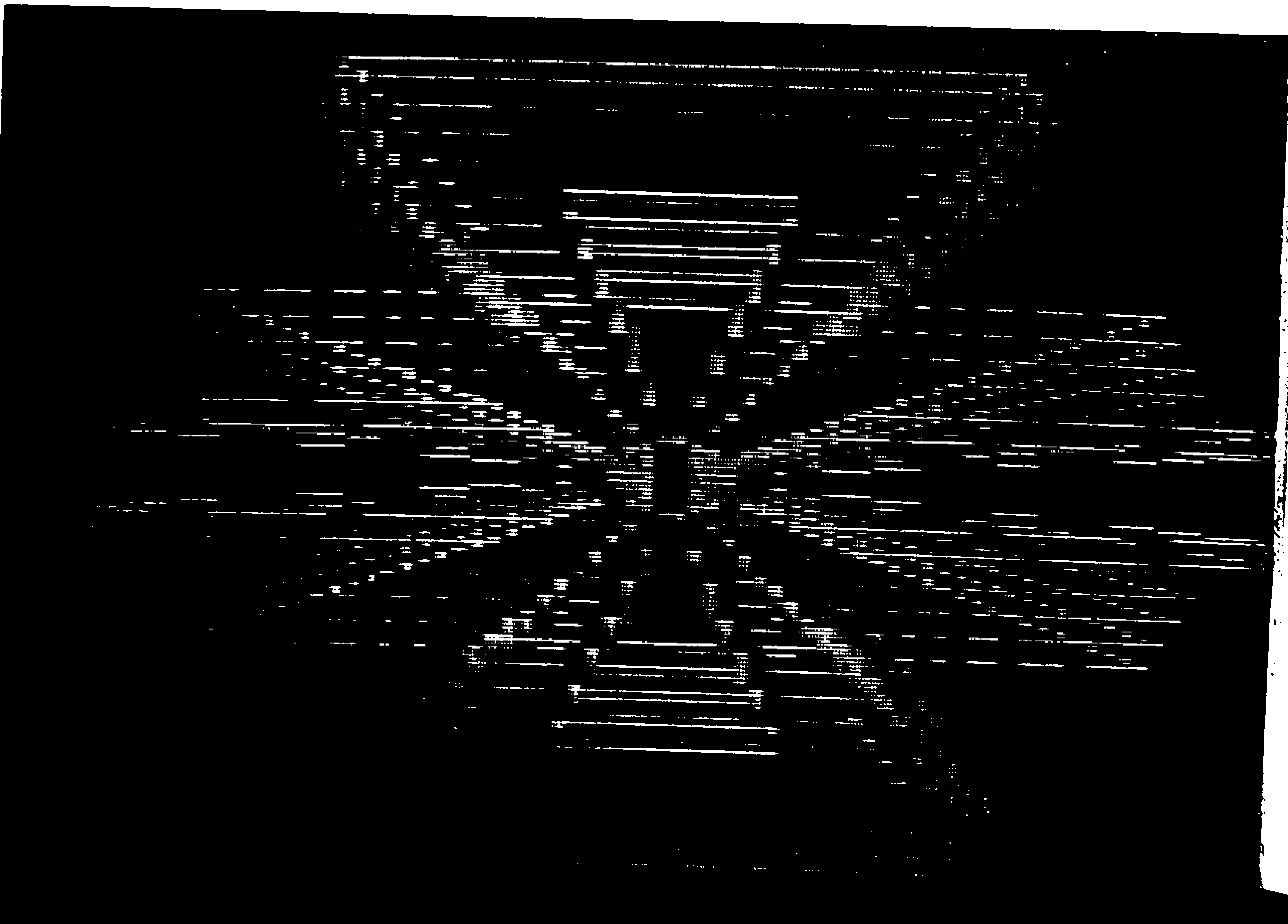
Een kubus programma *blz. 260*

Kaart van Nederland programma *blz. 129*



Graphics zwart/wit programma *blz. 51*

Graphics zwart/wit programma *blz. 51*



Om de grafiek geschikt te maken voor uw eigen gegevens, dient u in de regels 250-280 de woorden aan te passen.
In regel 20 moet u de waarden van A, B, C en D veranderen. (A=rood, B=geel, C=blauw, D=zwart)

Voortschrijdend gemiddelde

Een andere leuke toepassing van het maken van een grafiek in meer dan een kleur is het tekenen van een voortschrijdend gemiddelde van een grafiek. Het volgende programma tekent in rood een grafiek en vervolgens in geel het voortschrijdende gemiddelde.
Het tekenen van het voortschrijdend gemiddelde is een techniek om van een grafiek de ergste 'uitschieters' glad te strijken. Het gaat als volgt. Eerst wordt het gemiddelde genomen van de eerste vier gegevens:

gem = (ge1 + ge2 + ge3 + ge4) / 4

Vervolgens wordt het gemiddelde één gegeven verplaatst. Nu wordt het gemiddelde genomen van het tweede tot en met vijfde gegeven:

gem = (ge2 + ge3 + ge4 + ge5) / 4

Dit gaat net zo lang door tot alle gegevens gemiddeld zijn. Alle gemiddelden worden vervolgens weer door een lijn verbonden. Als de gegevens elke week opgenomen worden, dan heet een grafiek die telkens vier gegevens middelt een *vier-weeks-voortschrijdend gemiddelde*.
Het voordeel van deze methode is dat plotselinge wijzigingen in de grafiek slechts langzaam verwerkt worden in de grafiek van het voortschrijdende gemiddelde. Daardoor is de algemene ontwikkeling van de gegevens vaak beter afleesbaar aan de hand van het voortschrijdende gemiddelde dan aan de grafiek van alleen de gegevens zelf.

```
10 GRAPHICS 7+16
20 DIM Q(100)
30 DIM Z(100)
40 FOR X=1 TO 59
50 READ GE:Z(X)=GE
60 NEXT X
70 FOR X=5 TO 54
80 Q(X)=(Z(X-4)+Z(X-3)+Z(X-2)+Z(X-1)+Z(X)+Z(X+1)+Z(X+2)+Z(X+3)+Z(X+4))/8
90 NEXT X
100 COLOR 3
110 PLOT 0,0
120 DRAWTO 0,95
130 DRAWTO 159,95
140 PLOT 4,40
150 COLOR 1
160 FOR A=1 TO 59
170 DRAWTO A*2.3,95-Z(A)/1.7
180 NEXT A
190 COLOR 2
```

```
200 FOR T=0 TO 1
205 PLOT 9,30
210 FOR B=5 TO 54
220 DRAWTO B*2.3+T,95-Q(B)/1.7
230 NEXT B
235 NEXT T
240 GOTO 240
500 DATA 137,107,77,93,88,140,102,136,149,109,88,130
510 DATA 77,125,120,120,64,110,58
515 DATA 63,99,44,55,34,18,14,14,14
520 DATA 13,7,7,45,7,7,8,9,24,8,14,15,16,16,20,6,28,23
530 DATA 25,37,56,34,84,113,47,67,71,98,104,106,23
540 DATA 88,74,73,84,34,134,97,82
545 DATA 101,116,65,71,88,66,66,66
550 DATA 79,33,99,68,66,65,77,63,25
555 DATA 73,43,43,49,46,46,46,96,103,26
560 DATA 55,25,36,50,62,77,54,112,80,87,85
570 DATA 143,178,133,72,74,89,53,44,52,45,43,41,38
```

Regel 10 stelt het grafische scherm in en schakelt het tekstvenster uit. De regels 20 en 30 dimensioneren de ruimte voor een array. Door de gegevens in een array te plaatsen, gaat het tekenen van de grafiek veel sneller. De computer kan een array veel sneller uitlezen als de gegevens uit een datalijst.

Het inlezen van de array met gegevens uit de DATA-opdrachten vindt plaats in de regels 40-60.

De regels 70-90 zorgen verschillende malen voor het middelen van 8 waarden. Er is hier sprake van een voortschrijdend gemiddelde dat de gegevens per groep van 8 middelt.

De regels 100-130 tekenen het assenstelsel in blauw.

De regels 150-180 tekenen de gegevens uit de array Z(A) als een grafiek op het scherm. Om de grafiek mooi op het scherm te laten passen worden de gegevens in horizontale richting met 2.3 vermenigvuldigd. In de vertikale richting worden de gegevens afgetrokken van 95 (de hoogte van het scherm zonder tekstvenster) en gedeeld door 1.7. Door dit soort aanpassingen van de gegevens is het niet nodig de gemeten gegevens allemaal om te rekenen zodat ze op het scherm passen.

Door de omrekenfunktie aan de computer op te geven, verzorgt deze het omrekenen voor U. Op deze wijze kunt u andere getallen in de datalijst plaatsen, en met behulp van deze omrekenfactoren ervoor zorgen dat, ongeacht de grootte en de hoeveelheid van de gegevens, deze toch op het scherm passen. De regels 190-240 tekenen de grafiek van het voortschrijdend gemiddelde in geel. De grafiek wordt, door de factor T, twee maal getekend. De tweede keer is iets verschoven ten opzichte van de eerste keer. De lijn wordt daardoor een klein beetje dikker en daardoor nog nadrukkelijker aanwezig. De gegevens in de datalijst zijn willekeurig bij elkaar gezocht. U kunt de voor u belangrijke gegevens plaatsen. Bijvoorbeeld het gasgebruik per week. Of de buitentemperatuur elke week gemeten (elke week op dezelfde tijd, zelfde plaats), of de koersontwikkeling van de aandelen waarin u belegt, of de koers van de dollar, als u daarin speculeert.

17. TOETSENBORD EN JOYSTICK

Bij veel spelletjes, maar ook bij grafiek en andere toepassingen is het enorm handig als de computer snel kan reageren op signalen die de gebruiker wenst te geven. Het programma moet blijven doorlopen, ook in het geval de gebruiker tijdelijk geen signalen geeft. Het gebruik van de INPUT-opdracht is daarmee uitgesloten. Door deze opdracht wordt altijd gewacht met het programma totdat de RETURN-toets ingedrukt is. Heel handig als de computer juist moet wachten (de gebruiker moet de gelegenheid krijgen om een bladzijde te lezen en daarna een antwoord op een vraag te geven), maar zeer vervelend als het om een flitsend schietspel of een goed reagerend grafisch programma gaat. Het invoeren van direkte gegevens tijdens het lopen van het programma geschiedt meestal door het toetsenbord of door een joy-stick (in het nederlands ook wel een besturings-knuppel genoemd). Omdat dit de twee meest gebruikte vormen zijn, zullen we hier voorbeelden geven van programma's die de gebruiker de mogelijkheid geven om signalen door te geven aan de computer, terwijl het programma aan het lopen is.

Het toetsenbord

Om de werking van het toetsenbord te begrijpen is het nodig te weten op welke manier de Atari om gaat met de verschillende symbolen die er op de toetsen zijn aangegeven.
Op de toetsen zijn 26 letters, een stel cijfers, leestekens en grafische symbolen aangegeven. De computer maakt daarbij verschil tussen kleine letters en hoofdletters. Eerder in dit boek heeft u gelezen dat de computer elk teken een aparte code heeft toegekend. Een code die min of meer gestandaadiseerd is en ASCII-code heet. Deze ASCII-codes worden gebruikt door BASIC programma's.
Naast deze codering kent de Atari nog een andere codering waarmee elk letterteken aangegeven is. Dit is de interne code. Deze code wordt door het bedrijfssysteem van de Atari gebruikt om de karakters op het scherm te zetten. Elke code geeft de plaats van het karakter binnen de lijst met lettertekengegevens aan (zie bij het zelf maken van karakters). Het eerste karakter in de lijst in ROM is een spatie. De interne code daarvan is 0. De letter A is het drieendertigste teken van de lijst. De interne code is 33. Deze code wordt twee keer gebruikt. Als u een teken op het scherm wilt zetten, zet het bedrijfssysteem de ASCII-code van het teken om in de interne code. Deze interne code wordt vervolgens geplaatst in dat deel van het geheugen dat de opmaak van het beeldscherm bijhoudt. Op de plaats in het geheugen dat de gewenste positie op het beeldscherm aangeeft, wordt de interne code geplaatst van het teken dat daar neer gezet moet worden. De ANTIC chip leest 50 keer per seconde het deel van het geheugen dat over het beeldscherm gaat. De ANTIC komt de interne code tegen en gebruikt deze code om in de lijst met lettertekengegevens het patroon op te zoeken dat bij dit teken hoort, om vervolgens dat patroon door te geven aan het beeldscherm.

De derde code die de Atari gebruikt is de toetsenbord code. Dit is een getal dat door het toetsenbord afgegeven wordt als u een bepaalde toets indrukt. Bij elke (combinatie van) toets(en) hoort een andere toetsenbord code. Deze code wordt opgeslagen in geheugenlokatie 53769. De inhoud van deze lokatie wordt vervolgens door het bedrijfssysteem gekopieerd in lokatie 764. Aan die lokatie hebben wij als gebruikers veel meer, want dat is onderdeel van het RAM en daardoor te beinvloeden door middel van POKE. De waarde van lokatie 764 wordt gelezen en omgezet in een ASCII-code. Deze ASCII-code wordt vervolgens opgeslagen in geheugenlokatie 763.
Er is echter een nadeel verbonden aan de toetsenbordcodes. Ze hebben niets te maken met de ASCII-codes. *U vindt een complete lijst van toetsenbordcodes in de bijlagen.*

Nu we weten waar we de codes kunnen vinden die het toetsenbord afgeeft, kunnen we de codes gaan lezen in het programma.

```
10 GRAPHICS 11
20 COLOR 1
30 OPEN #1,4,0,"K:"
40 X=40:Y=80
50 IF PEEK(764)<>255 THEN GOSUB 500
60 PLOT X,Y
70 DX=0:DY=0
80 GOTO 50
500 GET #1,T
510 IF T=73 THEN DY=-1
520 IF T=77 THEN DY=1
530 IF T=74 THEN DX=-1
540 IF T=75 THEN DX=1
550 IF T=71 THEN COLOR 12
560 IF T=66 THEN COLOR 8
570 IF T=82 THEN COLOR 3
580 X=X+DX:Y=Y+DY:POKE 555,1
590 RETURN
```

Door de regels 10-30 wordt het scherm ingesteld, de kleur aangegeven en een IOCB kanaal voor invoer (taak nr. 4) via het toetsenbord (code K:) geopend. Regel 40 plaatst de grafische cursor in het midden van het scherm.
Regel 50 bekijkt geheugenplaats 764. Als daar de waarde 255 staat, is er geen toets ingedrukt. In dit geval hoeft de computer niets te doen. Het is echter niet de bedoeling dat het programma blijft wachten tot er wel een toets is ingedrukt. In dit programmavoorbeeld gaat het programma bij het niet ingedrukt zijn van een toets, verder met regel 60. Deze tekent een punt op de plaats van de grafische cursor. Na het terugzetten van de 'verandering' op nul in regel 70, gaat de computer terug naar regel 50, waar weer gevraagd wordt of er een toets is ingedrukt. In uw eigen programma kunt u natuurlijk het

programma op elke gewenste wijze verder laten gaan. Het gaat erom dat het programma bij het niet ingedrukt zijn van een toets, gewoon doorgaat. Alleen als er wel een toets is ingedrukt (dat wil zeggen dat de waarde in geheugenlokatie 764 een andere is dan 255), springt de computer tijdelijk naar een speciale subroutine om te handelen naar aanleiding van het toets indrukken.
In regel 500 neemt de computer via kanaal nummer 1, een waarde. Deze waarde wordt in de variabele T geplaatst.
De waarde die genomen wordt, is de toetsenbordcode. Door gebruik te maken van de GET opdracht krijgen we als waarde van T de ASCII code van de ingedrukte toets.

Als T=73 dan is de I ingedrukt. De verandering in de Y-richting wordt -1.
Als T=77 dan is de M ingedrukt. De verandering in de Y-richting wordt +1.
Als T=74 dan is de J ingedrukt. De verandering in de X-richting wordt -1.
Als T=75 dan is de K ingedrukt. De verandering in de X-richting wordt +1.
Als T=71 dan is de R ingedrukt. De kleur wordt rood (COLOR 12).
Als T=66 dan is de G ingedrukt. De kleur wordt groen (COLOR 8).
Als T=82 dan is de B ingedrukt. De kleur wordt blauw (COLOR 3).

Uit dit lijstje blijkt gelijk de gebruiksaanwijzing van het tekenbord dat wordt gevormd door het programma. Door één van de vier toetsen I, J, K of M in te drukken, bepaalt u de richting waarin een lijn wordt getrokken. Zolang u de toets ingedrukt houdt, beweegt de lijn. Door snel twee toetsen af te wisselen, kunt u diagonale lijnen, en na enige oefening ook kromme lijnen, tekenen. Door de R, G of B toets in te drukken, kunt u de kleur van de lijn beinvloeden. In de beginsituatie is de lijn licht oranje.
In regel 580 worden de coordinaten van de grafische cursor aangepast aan de hand van de veranderingsfactor die is gevonden door de GET-opdracht.
De laatste opdracht in regel 580 zorgt ervoor dat het zelf-herhalende karakter van het indrukken van een toets wat sneller gaat. Haal deze POKE 555,1 opdracht weg, en kijk hoeveel langzamer de toetsindruk zichzelf herhaalt.

Dit programma maakte gebruik van de ASCII codes en de GET opdracht. Hierboven werd verteld, dat de computer zelf gebruik maakt van een andere code om de toetsen aan te geven. Door gebruik te maken van die toetsen, kan het programma worden versneld. Het volgende programma is gelijk aan het bovenstaande, maar nu wordt de waarde van de ingedrukte toets direkt gelezen door middel van een PEEK opdracht.

```
10 GRAPHICS 11
20 COLOR I
30 X=40:Y=80
40 IF PEEK(764)<>255 THEN GOSUB 500
50 PLOT X,Y
60 DX=0:DY=0
70 GOTO 40
500 T=PEEK(764)
```

```
510 IF T=13 THEN DY=-1
520 IF T=37 THEN DY=1
530 IF T=1 THEN DX=-1
540 IF T=5 THEN DX=1
550 IF T=61 THEN COLOR 12
560 IF T=21 THEN COLOR 8
570 IF T=40 THEN COLOR 3
575 POKE 764,255
580 X=X+DX:Y=Y+DY:POKE 555,1
590 RETURN
```

Zoals u ziet heeft dit programma het openen van een IOCB kanaal niet meer nodig. De waarde wordt direkt in het geheugen gelezen.
Merk op dat de waarden waarmee T vergeleken wordt in de regel 510-570, andere waarden zijn, dan in het vorige programma. Nu worden de toetscodes gelezen in plaats van de ASCII codes.
Er is een nadeel verbonden aan het lezen van de toetscode in lokatie 764. Zolang er geen nieuwe toets wordt ingedrukt, blijft de oude waarde in 764 staan. Om te zorgen dat de lijn niet vanzelf blijft doorlopen, wordt in regel 575 de waarde van geheugenlokatie 764 weer gelijk gemaakt aan 255. Probeer uit wat er gebeurt als u deze regel weglaat.

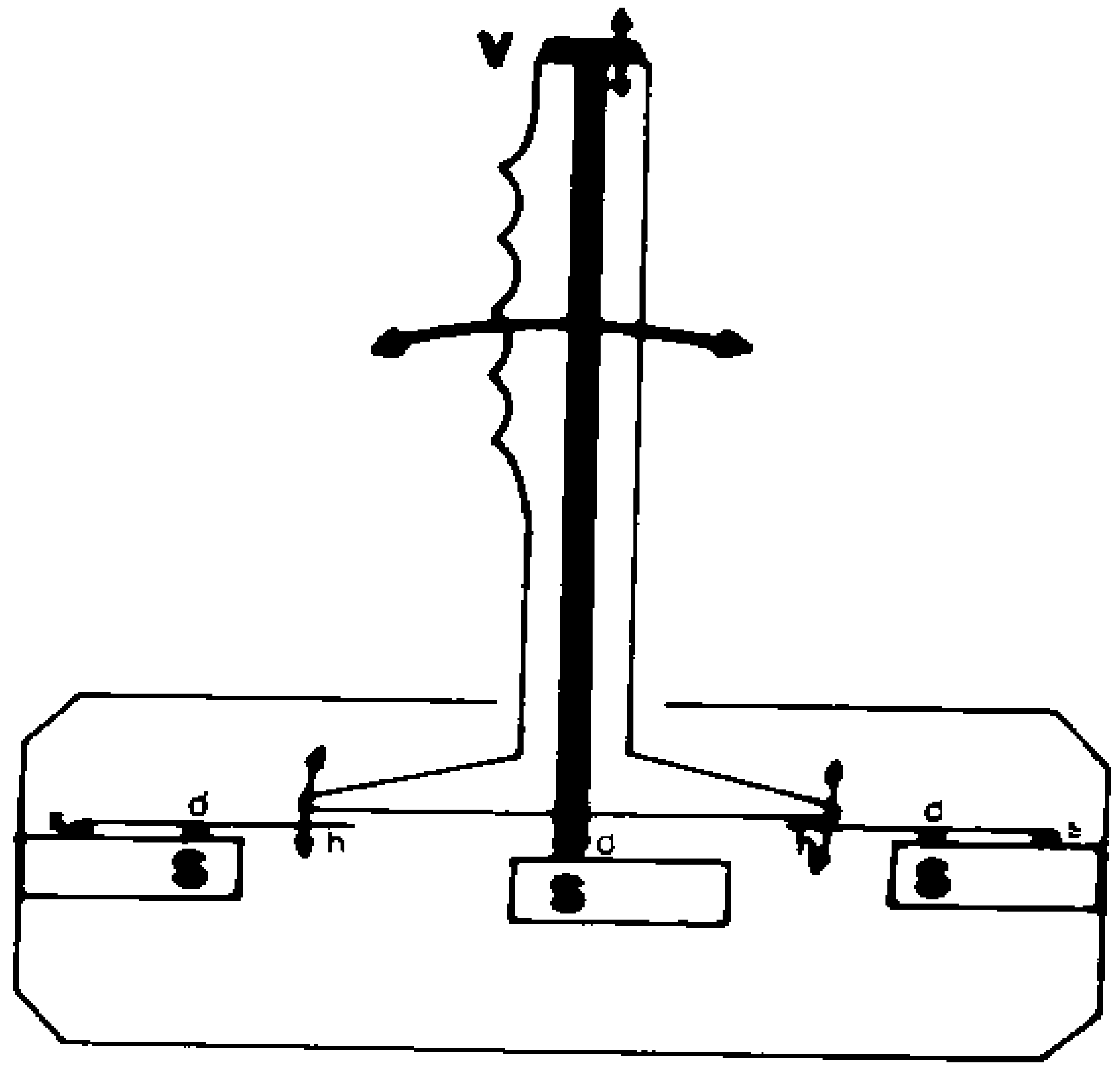
Joysticks

Het volgende onderdeel is alleen van belang voor hen die de beschikking hebben over minstens één joy-stick. Een joy-stick is een uiterst handig invoer-apparaat, en zeker bij spelletjes een onmisbare aanvulling van uw Atari. Heeft u nog geen joystick, leest u dit stuk dan vluchtig door, en besluit dan of u er misschien toch niet een kunt gebruiken.

Alle programma's in dit boek zijn te gebruiken met alleen een toetsenbord van de Atari als invoerinstrument. In de vorige twee programma's heeft u kunnen zien hoe u in een programma een subroutine kunt opnemen om bepaalde toetsindrukken snel af te lezen. Een joystick heeft als voordeel boven een toetsenbord dat hij gemakkelijker in de hand ligt en dat de richtingen voor het gevoel 'logischer' zijn. De eerste reactie als uw raket, laserwapen, kikker, raceauto of wat dan ook op het scherm naar rechts moet, is het naar rechts bewegen van de hand. Veel mensen reageren daardoor met een joystick veel sneller dan met een toetsenbord.

Het gebruik van joysticks komt meestal van pas bij het spelen van spellen. Door daarvoor het toetsenbord te gebruiken, veroorzaakt u snelle slijtage van bepaalde toetsen op het toetsenbord. Weliswaar slijt een joystick ook van veelvuldig gebruik, maar een joystick is gebouwd voor het (hardhandig) reageren op signalen van het beeldscherm en een joystick is veel goedkoper om te vervangen.

Een joystick is een rechtopstaande staaf die in verschillende richtingen geduwd kan worden. Door de joystick een bepaalde kant op te duwen, wordt er een kleine schakelaar omgehaald. Het stroompje dat daardoor gaat lopen, wordt door de computer afgelezen.
Omdat bijna alle joysticks verzegeld zijn door een sticker, die bij beschadiging de garantie ongeldig maakt, is het moeilijk om het binnenste van zo'n vernuftig apparaat te bekijken. Hieronder vindt u een vereenvoudigde tekening van het innerlijk van een joystick.



Dwarsdoorsnede:
V = Vuurknop
h = hefboom
S = Schakelaar
s = scharnier
d = drukpunt van schakelaar

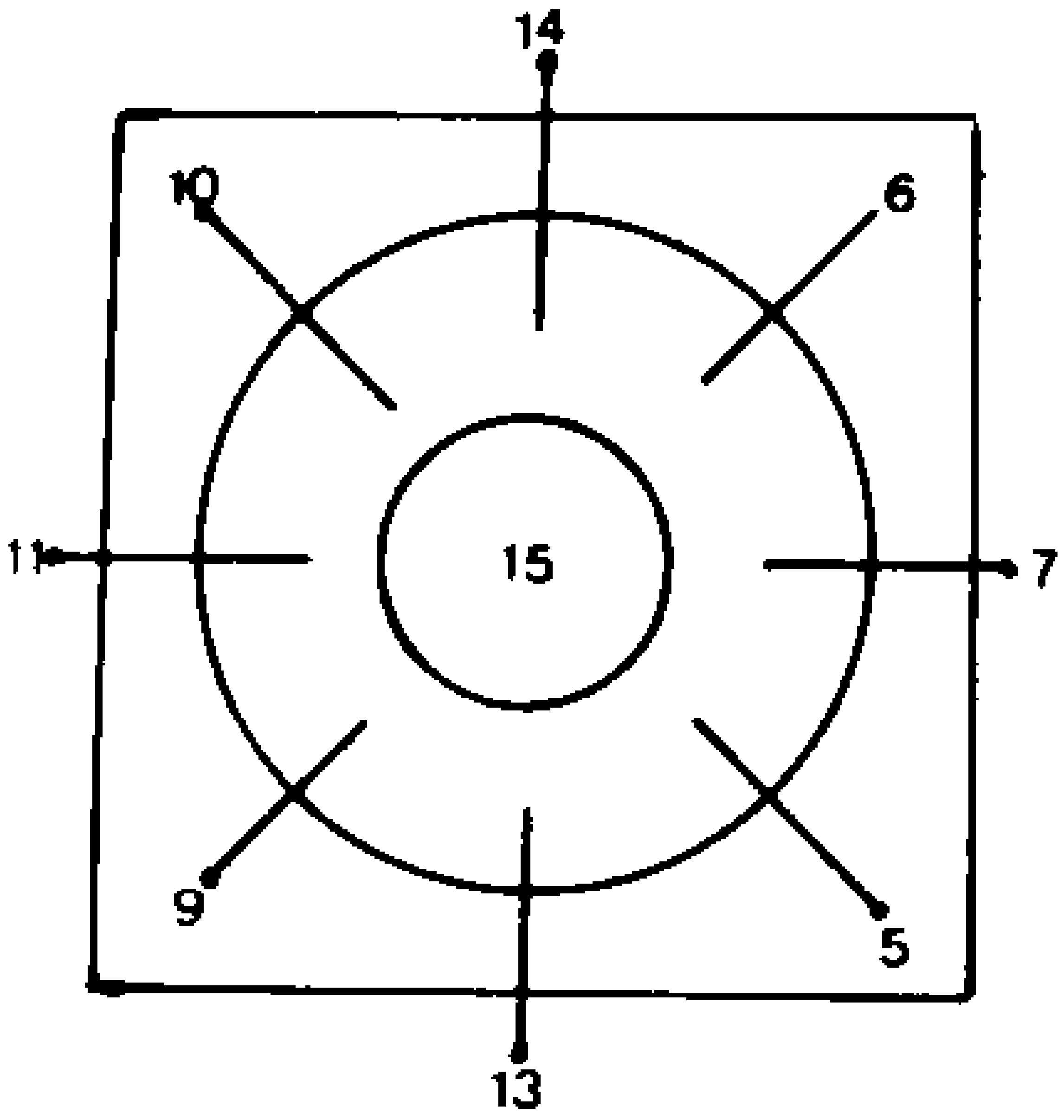
Het is niet belangrijk hoe de computer de signalen van de ingedrukte schakelaartjes omzet in gegevens voor het geheugen. Wel belangrijk is hoe de gebruiker in het geheugen kan opzoeken naar welke kant de joystick geduwd is.
Er zijn vier verschillende besturingsapparaten mogelijk. Deze zijn genummerd van 1 tot en met 4. De stand van elk apparaat wordt afgelezen door de opdracht

STICK(best.app.)

Achter STICK komt tussen haakjes het getal voor het besturingsapparaat. Deze lopen echter van 0 tot en met 3. De verhouding tussen deze twee is:

Besturingsapparaat 1 wordt gelezen door STICK(0)
Besturingsapparaat 2 wordt gelezen door STICK(1)
Besturingsapparaat 3 wordt gelezen door STICK(2)
Besturingsapparaat 4 wordt gelezen door STICK(3)

De waarden die de STICK opdracht af kan geven zijn afhankelijk van de stand van de joystick. Dit gaat volgens het volgende schema:



Als we deze informatie inbouwen in het tekenprogramma wordt dit:

```
10 GRAPHICS 11
20 KL=1:COLOR KL
30 X=40:Y=80
40 IF STICK(0)<>15 THEN GOSUB 500
45 IF STRIG(0)=0 THEN KL=KL+1:COLOR KL
50 PLOT X,Y
60 DX=0:DY=0
70 GOTO 40
500 T=STICK(0)
510 IF T=14 THEN DY=-1
520 IF T=10 THEN DY=-1:DX=-1
530 IF T=6 THEN DY=-1:DX=1
540 IF T=13 THEN DY=1
550 IF T=9 THEN DY=1:DX=-1
560 IF T=5 THEN DY=1:DX=1
570 IF T=11 THEN DX=-1
580 IF T=7 THEN DX=1
590 X=X+DX:Y=Y+DY
600 RETURN
```

In regel 10 wordt het grafische scherm geopend.
In regel 20 wordt de kleur ingesteld.
Regel 30 plaatst de grafische cursor in het midden van het scherm.
Regel 40 kijkt of de joystick in een andere stand dan de ruststand staat. Staat de joystick in de ruststand(dat wil zeggen 15), dan gaat het programma verder met de volgende regel. Als de joystick niet in de ruststand staat, wordt naar de subroutine gesprongen om de juiste stand af te lezen.

Regel 50 kijkt of de vuurknop ingedrukt is. Dit gaat door middel van de opdracht:

STRIG(best.app.)

Achter STRIG kunt u weer een getal van 0 tot en met 3 invullen. Zie het lijstje bij STICK.
DE STRIG opdracht geeft een 0 als waarde als de vuurknop is ingedrukt en een 1 als de knop niet is ingedrukt.
Door regel 45 wordt de waarde die de kleur van de te tekenen lijn bepaalt, met één verhoogd als de vuurknop ingedrukt wordt.
Regel 50 tekent het volgend punt van de lijn.
Regel 60 stelt de veranderingen weer op nul.
Regel 70 springt terug om opnieuw de joystick af te lezen.
In regel 500 wordt de waarde van T gelijk gemaakt aan de afgelezen waarde van de joystick.
De regels 510-580 kijken welke waarde T heeft, en bepalen aan de hand daarvan of de waarde van de X- en Y-coordinaat verhoogd of verlaagd moet worden. Voor de gebruikte waarden: zie tekening met joy-stick standen.
Regel 590 zorgt ervoor dat de X- en Y-coördinaat worden aangepast.
Door regel 600 wordt uit de subroutine terug gesprongen naar het hoofdprogramma.

Het bovenstaande programma geeft duidelijk weer hoe een subroutine gebruikt kan worden om de stand van de joystick af te lezen. U kunt deze joystickroutine in uw eigen programma's opnemen, om zodoende de programma's geschikt te maken voor het gebruik met een joystick.
Er zijn echter vele verschillende manieren om een subroutine te maken die de stand van de joystick afleest. Bovenstaande methode is wel duidelijk, maar tamelijk langzaam. De volgende methode is veel korter.

```
10 GRAPHICS 11
20 KL=1:COLOR KL
30 X=40:Y=80
40 IF STICK(0)<>15 THEN GOSUB 500
45 IF STRIG(0)=0 THEN KL=KL+1:COLOR KL
50 PLOT X,Y
60 DX=0:DY=0
70 GOTO 40
500 T=STICK(0)
510 DX=(T=5 OR T=6 OR T=7)-(T=9 OR T=10 OR T=11)
520 DY=(T=5 OR T=9 OR T=13)-(T=6 OR T=10 OR T=14)
530 X=X+DX:Y=Y+DY
540 RETURN
```

Een truuk om subroutines nog sneller te laten werken, is door ze niet zoals gebruikelijk aan het einde van een programma te zetten, maar aan het begin. Elke keer als de computer naar een subroutine gaat, hoeft hij niet telkens alle regelnummers af te tellen tot hij bij de juiste regel is. Normaal gesproken is het erg lelijk om dit in een programma te doen, omdat het de leesbaarheid van een programma sterk vermindert. Het aflezen van een joystick moet echter voor alles snel gebeuren, daar het anders de snelheid van het programma ernstig vertraagt.
In het volgende programma is de joystick routine naar het begin van het programma verplaatst, en is in de regels 50-80 een beveiliging ingebouwd tegen 'van het scherm lopen'. Probeert de gebruiker een lijn te trekken buiten het toegestane gebied, dan wordt de lijn aan de andere kant van het scherm voortgezet. Probeert u aan de onderkant te ver door te tekenen, dan verschijnt de lijn bovenin het scherm weer en vice versa. Hetzelfde geldt links-rechts en rechts-links.

```
10 GOTO 100
20 T=STICK(0)
30 DX=(T=5 OR T=6 OR T=7)-(T=9 OR T=10 OR T=11)
40 DY=(T=5 OR T=9 OR T=13)-(T=6 OR T=10 OR T=14)
50 X=X+DX:IF X>79 THEN X=0
60 IF X<0 THEN X=79
70 Y=Y+DY:IF Y>191 THEN Y=0
80 IF Y<0 THEN Y=191
90 RETURN
100 GRAPHICS 11
110 KL=1:COLOR KL
120 X=40:Y=80
130 IF STICK(0)<>15 THEN GOSUB 20
140 IF STRIG(0)=0 THEN KL=KL+1:COLOR KL
150 PLOT X,Y
160 DX=0:DY=0
170 GOTO 130
```

Raketspel

Als intermezzo een spelletje waarin de technieken van het vorige en dit hoofdstuk verenigd zijn.
U bent de bestuurder van een ruimteschip. U moet de redder van de wereld worden. Allerlei geniepige snodaards uit het verre heelal schieten raketten op u af (een zelf-maak-karakter). U heeft de beschikking over een ruimteschip (drie zelf-maak-karakters) waarmee u de raketten kunt opvangen. Door middel van de joystick kunt u het ruimteschip naar links en rechts bewegen. Voor elke gevangen raket krijgt u punten. Mist u een raket? Geen nood. Op de grond aangekomen, blijken de raketten samen leuke bloemen (vingerhoedskruid) te vormen. 10 tellen....

```
1 GOTO 10
2 REM JOYSTICK ROUTINE
3 S=STICK(0)
4 DX=(S=5 OR S=6 OR S=7)-(S=9 OR S=10 OR S=11)
5 X=X+DX:IF X>33 THEN X=33
6 IF X<1 THEN X=1
9 RETURN
10 POKE 106,PEEK(106)-4:GRAPHICS 0
20 POSITION 10,20:PRINT "EVEN GEDULD A.U.B."
25 FOR WA=1 TO 300:NEXT WA
30 BEGINLIJST=PEEK(106)*256
40 POKE 756,BEGINLIJST/256
50 FOR KOPIE=0 TO 1023
60 POKE BEGINLIJST+KOPIE,PEEK(57344+KOPIE)
70 NEXT KOPIE
80 REM
90 FOR Q=33 TO 37
100 FOR A=0 TO 7
110 READ GE
120 POKE BEGINLIJST+((Q-32)*8)+A,GE
130 NEXT A
140 NEXT Q
200 REM SPEL
210 GRAPHICS 0:SETCOLOR 1,0,14
220 SETCOLOR 2,13,0:REM GELE TEKENS
230 SETCOLOR 4,8,6:REM RAND DONKERBLAUW
240 POKE 756,BEGINLIJST/256:REM WIJZER NWE LETTEK.
250 POKE 752,1:REM CURSOR UIT
260 DIM A(9):DIM Z(9)
265 FOR Q=1 TO 9:F=0:A(Q)=F:Z(Q)=F:NEXT Q
270 SCORE=0
280 X=20
290 POSITION 0,22:FOR Q=1 TO 39:PRINT CHR$(33);:NEXT Q
300 GOSUB 500
310 GOSUB 600
320 GOTO 310
330 END
499 REM STERROUTINE
500 FOR S=50 TO 0 STEP -1
510 P=INT(RND(0)*38)+1:Q=INT(RND(0)*22)
520 POSITION P,Q:PRINT ".";
530 POSITION 2,2:PRINT S;" ";
540 NEXT S
550 POSITION 2,2:PRINT "   ";
560 RETURN
599 REM VERPLAATSROUTINE
600 B=INT(RND(0)*7)+1
```

```
610 POSITION 35,2:PRINT SCORE;
620 LR=4*B
630 Z(B)=1
640 POSITION LR,Z(B):PRINT CHR$(34);
650 POSITION LR,Z(B)-1:PRINT " ";
660 REM EVENTUELE PAUZE
670 GOSUB 2
672 POSITION X,15:PRINT CHR$(32);CHR$(35);
673 PRINT ;CHR$(36);CHR$(37);CHR$(32);
674 IF Z(B)=15 AND ABS(X+2-LR)<=1 THEN GOSUB 900
680 Z(B)=Z(B)+1
690 IF Z(B)<22-A(B) THEN GOTO 640
699 REM VERPLAATSROUTINE
700 A(B)=A(B)+1
710 IF A(B)=5 THEN POSITION 10,22:? "DE MAAT IS VOL";:END
720 SOUND 0,A(B)*10,14,10
725 FOR WA=1 TO 9:NEXT WA:SOUND 0,0,0,0
730 RETURN
899 REM ROUTINE VOOR RAAK
900 SCORE=SCORE+10
910 SOUND 0,50,12,8:FOR WA=1 TO 19:NEXT WA
920 SOUND 1,100,12,10:FOR WA=1 TO 19:NEXT WA
930 SOUND 2,150,14,12:FOR WA=1 TO 19:NEXT WA
940 SOUND 3,200,14,12:FOR WA=1 TO 9:NEXT WA
950 SOUND 0,0,0,0:SOUND 1,0,0,0
955 SOUND 2,0,0,0:SOUND 3,0,0,0
960 GOTO 310
1000 DATA 85,170,85,170,255,255,255,255
1010 DATA 153,153,90,90,60,60,24,24
1020 DATA 0,6,1,0,128,255,127,7
1030 DATA 66,36,153,126,255,255,255,195
1040 DATA 0,96,128,0,1,255,254,224
```

Veel plezier met dit spannende spel. U kunt het programma op verschillende wijzen verfraaien.

U kunt in plaats van een joystick besturing zorgen voor besturing door het toetsenbord. Verander daarvoor de eerste regels van het programma. U kunt de vorm van de raketten veranderen, of er voor zorgen dat er steeds andere raketten naar beneden komen vallen. U kunt ervoor zorgen dat de vliegende schotel alle kanten op kan, waarbij de bestuurder meer punten krijgt als hij een raket hoger op het scherm vangt. U kunt een fraaie bladzijde maken voor de tijd dat de gebruiker moet wachten op het kopieren van de letterteken lijst. Of een mooie bladzijde als de speler klaar is met het spel.

18. GRAFIEK: HOGE RESOLUTIE

In de vorige hoofdstukken heeft u geleerd hoe u kleurrijke tekeningen kunt maken. Hoe meer kleur, hoe minder fijn de lijnen zijn. Dit is niet altijd een nadeel, zoals u hiervoor al kon zien, maar soms is het noodzakelijk om een tekening met fijne lijnen te maken.
Hoe dun de lijnen op uw Atari kunnen zijn, is afhankelijk van het type. Hoe dunner de lijnen, hoe meer geheugenruimte het opbouwen van het scherm vraagt. Dit zorgt ervoor dat bepaalde grafische schermen op enkele Atari's niet mogelijk zijn. Past een bepaald voorbeeld uit dit hoofdstuk niet op uw Atari, vervang dan de opdracht die het grafische scherm instelt voor een opdracht die een scherm kiest dat minder geheugenruimte vraagt.
Vervang in datzelfde programma dan ook de factoren die bepalen waar de verschillende lijnen worden getrokken. De Atari staat u niet toe om buiten het scherm te tekenen.

Stippen

Het scherm met het fijnste oplossende vermogen is zonder twijfel scherm 8. *Alle voorbeelden in dit hoofdstuk zijn bedoeld voor scherm 8.* Wilt u echter de tekeningen met wat meer kleur zien, dan kunt u ook gebruik maken van bijvoorbeeld scherm 7. Dit scherm heeft echter een vier keer zo grof oplossend vermogen.
De opdrachten die gebruikt kunnen worden voor het maken van grafische opdrachten, zijn:

GRAPHICS
PLOT
DRAWTO
COLOR
SETCOLOR
LOCATE

Het volgende programma laat het gebruik van de PLOT opdracht zien. U kunt met deze opdracht een stip op elke plaats op het scherm laten verschijnen. Daardoor kunnen zeer fraaie tekeningen worden gevormd. De gebruiker kan bijna alles wat in zijn/haar fantasie opkomt op het scherm tekenen. Het programma tekent op willekeurige plaatsen stippen. De stippen beginnen echter in het midden van het scherm en zullen daar omheen blijven cirkelen.

```
10 GRAPHICS 8+16
15 SETCOLOR 2,13,0
20 SETCOLOR 1,13,14:COLOR 1
30 PLOT 160,80
40 X=0:Y=0:DX=0:DY=0
50 X=INT(RND(0)*10-4.5)
60 Y=INT(RND(0)*10-4.5)
```

```
70 DX=DX+X:DY=DY+Y
80 PLOT 160+DX,80+DY
90 GOTO 50
```

Regel 10 stelt het grafische scherm in en verwijdert het tekstvenster.
Regel 15 maakt de achtergrond zeer donker geel.
Regel 20 maakt de kleur van de stippen heldergeel.
Regel 30 tekent een punt in het midden van het scherm.
De regels 30-70 zorgen voor een wisselende waarde van DX en DY. Regel 80 tekent een stip.

Lijnen

Met behulp van de PLOT opdracht kunnen behalve stippen, ook lijnen getekend worden. Een lijn is immers niets anders dan een verzameling stippen. Probeer het volgende programma uit:

```
10 GRAPHICS 8+16
15 SETCOLOR 2,13,0
20 SETCOLOR 1,13,14:COLOR 1
30 FOR X=1 TO 250
40 PLOT X,100
50 NEXT X
60 GOTO 60
```

In dit programma wordt X van 1 tot 250 telkens met 1 verhoogd. Op het punt (X,100) wordt vervolgens een stip gezet. Door op deze wijze een heleboel stippen achter elkaar te zetten, krijgt u een rechte lijn.
Op precies dezelfde wijze kunt u ook een lijn maken die van boven naar beneden loopt. U hoeft dan alleen de Y-coordinaat te laten veranderen door een FOR...NEXT lus. Om een diagonale lijn te krijgen verandert u allebei de coordinaten. Bijvoorbeeld:

```
10 GRAPHICS 8+16
15 SETCOLOR 2,13,0
20 SETCOLOR 1,13,14:COLOR 1
30 FOR X=1 TO 250
40 PLOT X,X/2
50 NEXT X
60 GOTO 60
```

In dit programma worden zowel de X- als de Y-coordinaat veranderd. Daarvoor wordt in een FOR..NEXT lus de X-coordinaat van 1 tot 250 verhoogd en wordt de Y-coordinaat telkens gelijk gemaakt aan de helft van de X-coordinaat. De diagonale lijn die u nu te zien krijgt is dus een grafische weergave van de funktie Y=X/2. *(de grafische afbeelding staat ondersteboven doordat de Atari de Y-as van boven naar beneden laat lopen in plaats van het meer gebruikelijke, van onder naar boven)*

Er is een veel snellere methode om lijnen te trekken. Deze bent u al eerder tegen gekomen. Het is de opdracht:

DRAWTO

Achter de DRAWTO opdracht worden de coordinaten van het eindpunt van de lijn opgegeven. Het beginpunt van de lijn is de plaats van de grafische cursor, op het moment dat de opdracht uitgevoerd gaat worden. De plaats van de grafische cursor is te beinvloeden door de PLOT opdracht. Het programma hiervoor zou herschreven kunnen worden tot:

```
10 GRAPHICS 8+16
15 SETCOLOR 2,13,0
20 SETCOLOR 1,13,14:COLOR 1
30 PLOT 1,0
40 DRAWTO 250,125
60 GOTO 60
```

Het programma is hierdoor korter geworden. Belangrijker is echter dat de lijn vele malen sneller wordt getrokken.

Krommen

Met een verzameling punten is het niet alleen mogelijk om rechte lijnen samen te stellen, maar ook om kromme lijnen te vormen.
Een kromme lijn kan worden gezien als een verzameling punten of als een verzameling van zeer korte rechte lijnstukjes die allemaal onder een hoek ten opzichte van elkaar staan. De eenvoudigste manier om een gekromde lijn te krijgen, is door een verzameling punten te tekenen.

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,14
30 SETCOLOR 1,0,0:COLOR 1
40 FOR X=0 TO 185
50 Y=INT(X*X/200)
60 PLOT X,Y
70 NEXT X
80 GOTO 80
```

Dit programma tekent, in zwart op wit, de grafiek van de funktie Y=X*X. Eigenlijk tekent het programma de funktie Y=X*X/200. De deling door 200 is bedoeld om de grafiek op een beetje redelijke wijze op het scherm te laten passen. Wordt een dergelijke schaalaanpassing niet gebruikt dan zou de grafiek als een bijna rechte lijn naar beneden gaan. De grafiek staat ondersteboven. Wilt u dit veranderen (dit geldt voor alle programma's in dit hoofdstuk) dan moet u voor de Y-coordinaat niet de waarde Y geven, maar de waarde **191-Y**. Het probleem bij de grafiek is dat er weliswaar een schitterende kromme lijn uit

komt, maar dat deze onderaan (of bovenaan) uit elkaar dreigt te vallen, doordat de punten te ver uit elkaar komen te staan. Dit kan verbeterd worden door geen punten te zetten, maar lijnen te trekken. Verander regel 60 in:

```
60 DRAWTO X,Y
```

en voeg toe:

```
35 PLOT 0,0
```

Bij de getekende grafiek ontbreekt dat deel van de grafiek dat de waarden voor X kleiner dan nul aangeeft. Het is immers niet mogelijk om buiten het scherm te tekenen. De Atari kent een scherm waarvan de kolommen van 0 tot en met 319 lopen (in scherm 8). Om nu toch de hele grafiek, dat wil zeggen ook het deel met negatieve X-waarden, te laten zien, kan er gebruik gemaakt worden van schaalaanpassing en verschuiving. U schuift de grafiek als het ware een beetje naar rechts. U doet daarbij net alsof het nulpunt (0,0) meer naar rechts ligt. Het resultaat wordt:

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,14
30 SETCOLOR 1,0,0:COLOR 1
35 PLOT 0,190-INT(-125*-125/100)
40 FOR X=-125 TO 125
50 Y=INT(X*X/100)
60 DRAWTO X+125,190-Y
70 NEXT X
80 GOTO 80
```

Stuiterbal

Een aardige toepassing van deze technieken is het volgende programma dat de loop van een stuiterbal simuleert. Een bal wordt voorwaarts weggeworpen.

```
10 GRAPHICS 0
20 BX=11:BY=50
30 VS=0:N=0
40 ? :? :? "Dit prog. laat de loop van een"
50 PRINT "stuiterende bal zien."
60 PRINT :PRINT
70 ? "Horizontale beginsnelheid (0-25)"
80 INPUT HS
110 GRAPHICS 8+16
120 SETCOLOR 2,0,14
130 SETCOLOR 1,3,0:COLOR 1
140 PLOT 10,10
150 DRAWTO 10,185:DRAWTO 310,185
```

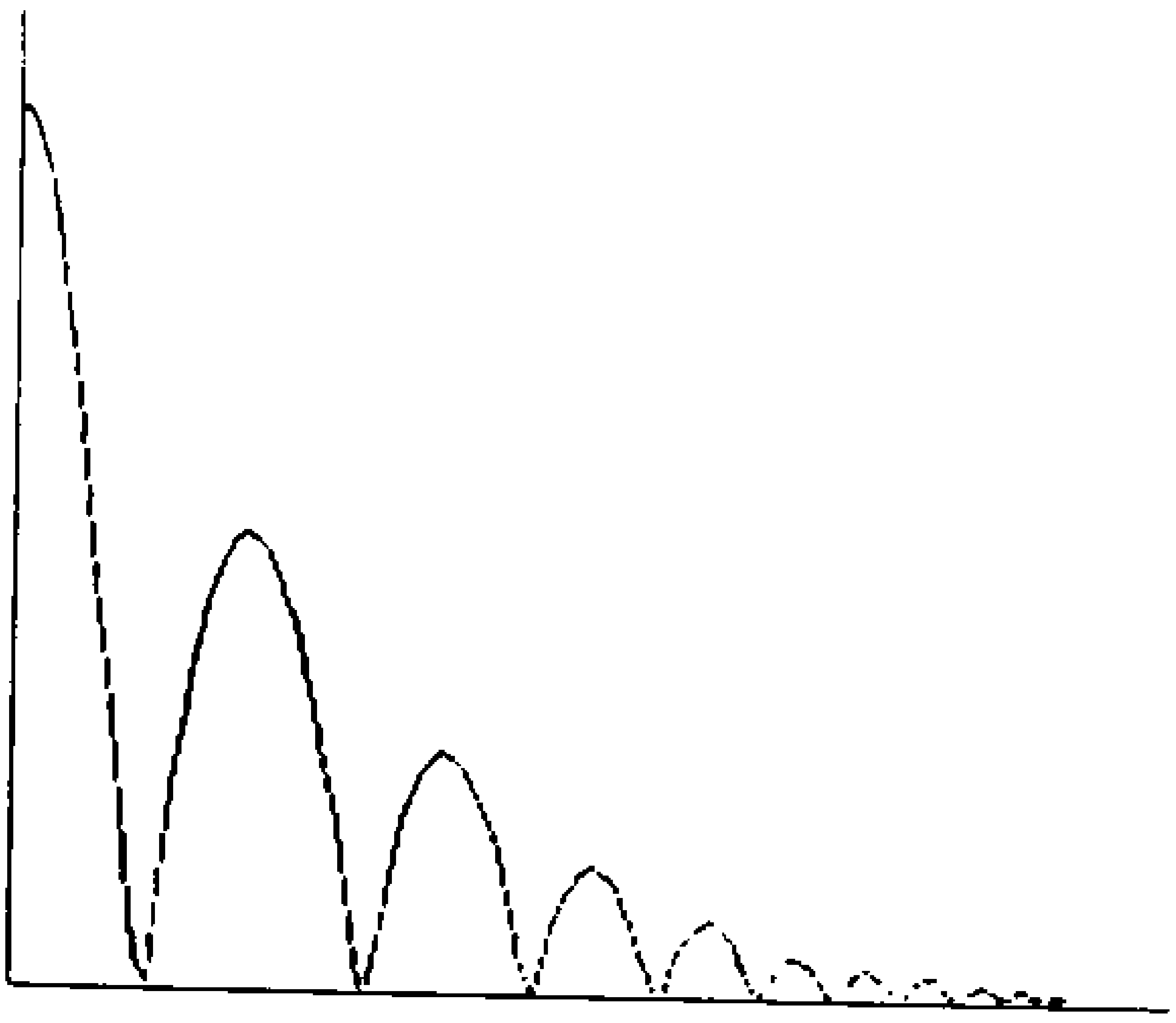
```
155 DRAWTO 310,10
160 PLOT BX,BY
199 REM LUS
200 BX=BX+HS/2
210 VS=VS+2
220 BY=BY+VS
230 IF BY>=184 THEN GOSUB 500
240 IF BX>=309 THEN GOSUB 1000
250 IF BX<=11 THEN GOSUB 1500
300 DRAWTO BX,BY
310 FOR WA=1 TO (190-BY)/5:NEXT WA
320 IF N<=20 THEN GOTO 200
330 GOTO 330
340 END
499 REM GROND
500 BY=184
510 VS=-0.9*VS
520 N=N+1
530 SOUND 1,100,14,12:FOR WA=1 TO 3
535 NEXT WA:SOUND 1,0,0,0
540 RETURN
999 REM MUUR RECHTS
1000 BX=309
1010 HS=-0.9*HS
1020 SOUND 1,100,14,12:FOR WA=1 TO 3
1025 NEXT WA:SOUND 1,0,0,0
1030 RETURN
1499 REM MUUR LINKS
1500 BX=11
1510 HS=-0.9*HS
1520 SOUND 1,100,14,12:FOR WA=1 TO 3
1525 NEXT WA:SOUND 1,0,0,0
1530 RETURN
```

De eerste regels dienen voor het kiezen van de schermmodus en voor het instellen van de verschillende variabelen. De variabelen BX en BY geven de positie van de bal aan. VS is de vertikale snelheid, HS is de horizontale snelheid en N is het aantal keren dat de bal heeft gestuiterd. De horizontale snelheid kan door de gebruiker ingesteld worden door regel 70 en 80.
In de regels 110-150 wordt er een grafisch scherm ingesteld en worden de kaders getrokken. Daarna wordt de grafische cursor klaar gezet op het beginpunt.

Regel 200 bepaalt de X-coordinaat van de bal. Deze is telkens gelijk aan de oude plaats plus de helft van de door de gebruiker ingevoerde horizontale snelheid.

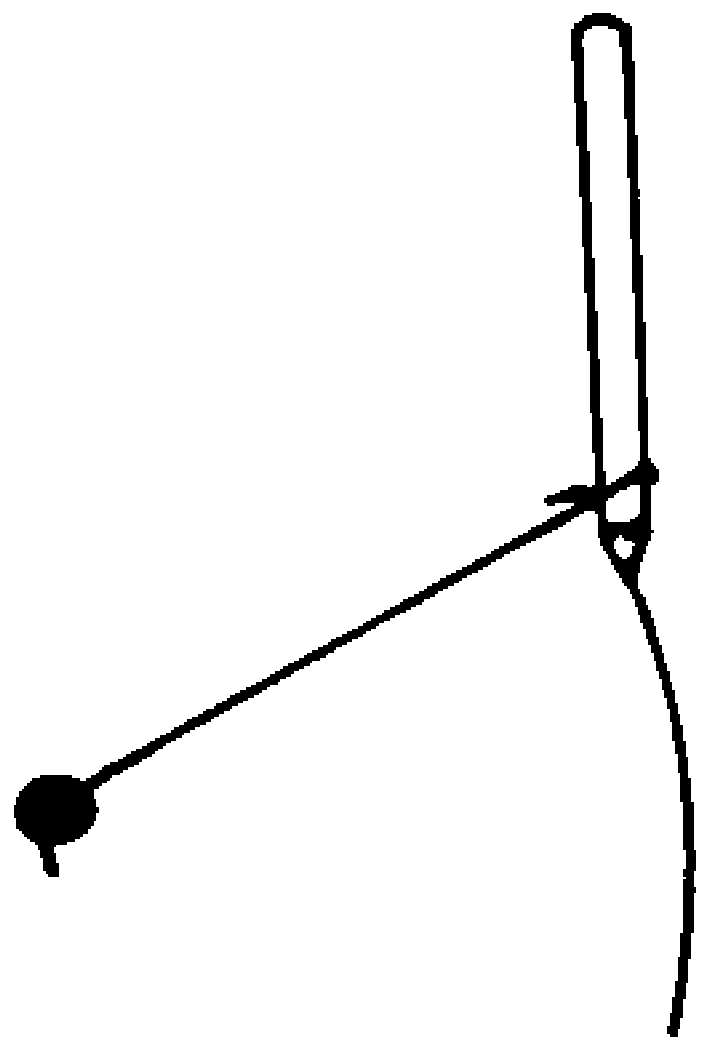
De vertikale snelheid neemt toe (door de zwaartekracht). De vertikale snelheid is telkens gelijk aan de oorspronkelijke snelheid plus 2 (regel 210). De vertikale

plaats (de Y-coordinaat) is afhankelijk van de vorige plaats en van de vertikale snelheid (regel 220).
In de regels 230, 240 en 250 kan naar aparte subroutines worden gesprongen. Dit is afhankelijk van de plaats van de bal. Bevindt de bal zich vlak bij de grond dan gaat de computer naar regel 500. Voor een botsing met de linker- of rechtermuur gaat de computer naar regel 1000 of 1500.
Regel 300 zorgt voor het tekenen van de baan van de bal.
Regel 310 zorgt voor een vertraging. De vertraging is afhankelijk van de hoogte van de bal. Hoe hoger de bal is, hoe langzamer de bal gaat. Ook dit is bedoeld om een betere simulatie te geven.
Regel 320 zorgt voor een beperking van het aantal stuiters. Regel 330 zorgt ervoor dat het grafische scherm niet bij het einde van het programma verdwijnt.
De subroutine van regel 500 zorgt ervoor dat de bal precies de grond raakt (BY=184) en keert de richting van de snelheid om (VS=-.9*VS). Daarbij wordt de snelheid een beetje verkleind, omdat een stuiterbal bij het raken van een object altijd een deel van zijn snelheid verliest (de snelheid wordt omgezet in warmte-energie. Voelt u het verschil in temperatuur maar eens tussen een nog niet gebruikte en een paar minuten gebruikte squashbal. Wilt u de baan van een slechte stuiterbal zien, dan moet u in regel 510 de factor .9 verkleinen. Een absolute stuiterbal heeft een verliesfactor van 1.
De subroutine van regel 1000 is vergelijkbaar met de subroutine van regel 500. Nu wordt echter het stuiteren tegen de rechtermuur behandeld. De horizontale snelheid wordt omgedraaid en een beetje verkleind. De subroutine van regel 1500 is gelijk aan die van regel 1000, maar dan voor de linkermuur.

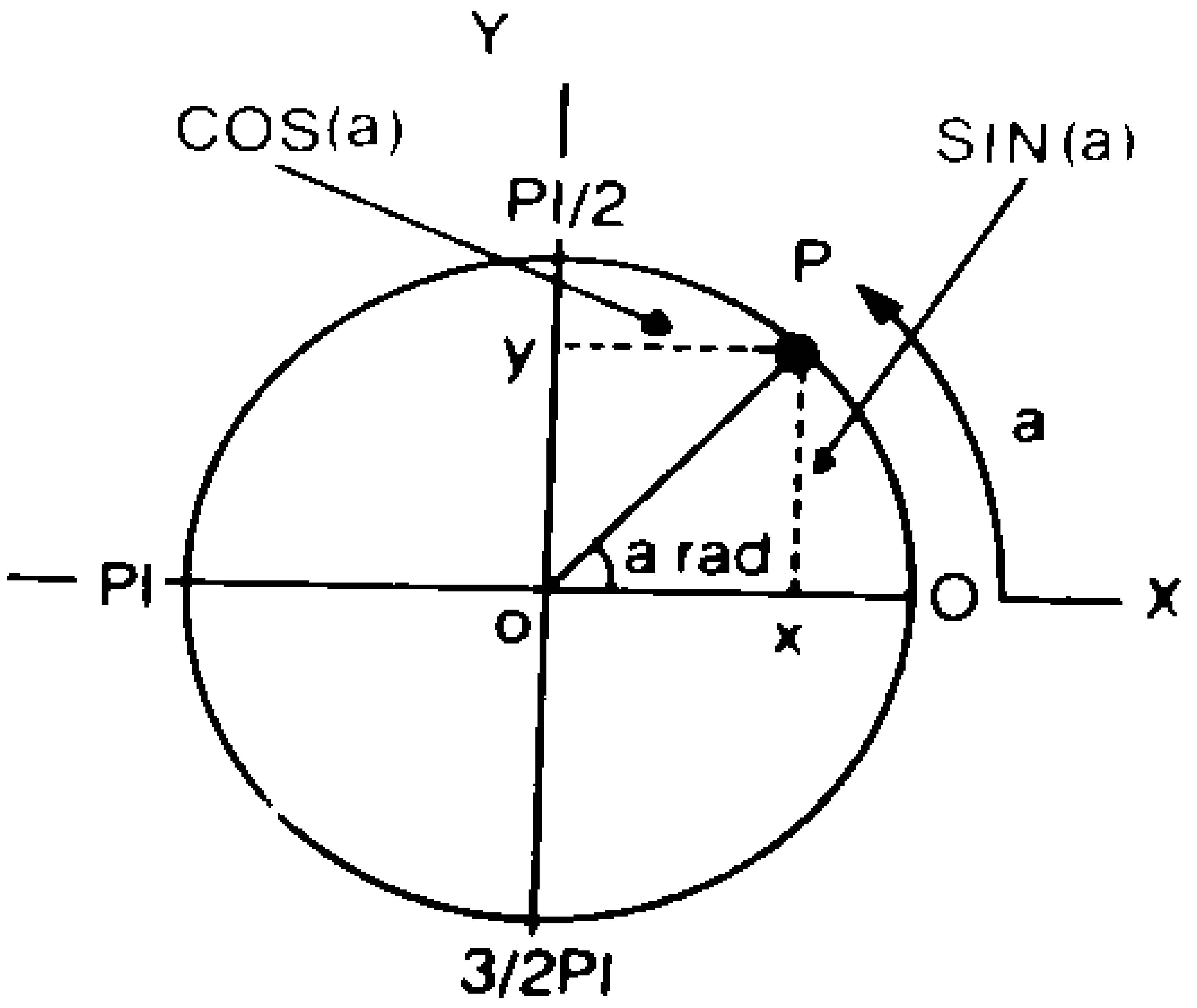


Cirkels en aanverwanten

Er is een speciale kromme lijn die altijd bijzonder tot de verbeelding spreekt. Het is de kromme lijn die geen begin of einde heeft: de cirkel. De Atari kent geen speciale opdracht om cirkels te tekenen, maar met een klein beetje wiskunde is dat geen bezwaar.
Een cirkel kan getekend worden door een touwtje te nemen en aan het einde van dat touwtje een potlood te bevestigen. Het andere eind van het touwtje wordt op een vast punt geplaatst, bijvoorbeeld door een punaise. Door het touwtje strak te houden en 360 graden rond te draaien, tekent het potlood een cirkel:



Op dezelfde wijze kunnen we de computer een lijn laten trekken. We moeten daarvoor elk punt dat het potlood tekent, aan kunnen geven in een X-coordinaat en een Y-coordinaat. Daartoe trekken we twee assen door het middelpunt van de cirkel:



Van elk punt op de cirkel is nu gegeven dat de hoogte ten opzichte van de horizontale as gelijk is aan de sinus van de hoek die 'het touwtje' (= de rechte lijn tussen het punt en het middelpunt van de cirkel maakt) maal de straal van de cirkel (de straal is de helft van de lengte van de doorsnede van de cirkel). De afstand links-rechts ten opzichte van de verticale as door het middelpunt is gelijk aan de cosinus van de hoek maal de straal van de cirkel. Kortom; een mond vol. Maar het geeft ons de mogelijkheid een mooie ronde cirkel te tekenen:

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,14
30 SETCOLOR 1,0,0:COLOR 1
40 FOR HOEK=0 TO 360
50 PLOT 160+COS(HOEK)*40,90+SIN(HOEK)*40
60 NEXT HOEK
70 GOTO 70
80 GOTO 80
```

Run dit programma en zie dat er inderdaad eem cirkel ontstaat, maar wel op een vreemde manier. Eerst tekent de computer een paar punten ver van elkaar en langzaam maar zeker worden de andere punten ertussen opgevuld. De reden voor deze werkwijze van de computer is gelegen in de manier waarop hoeken aangeduid worden.
Een hoek kan aangeduid worden in graden of in radialen. Een cirkel gaat 360 graden of twee maal pi radialen rond. Pi is een griekse letter die een getal aangeeft dat ongeveer 3.14 groot is. De precieze waarde van pi, kent de Atari niet, maar u kunt het getal zo precies mogelijk benaderen door de wiskundige berekening:

```
4 * ATN(1)
```

Als we dit verwerken in ons programma krijgen we:

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,14
30 SETCOLOR 1,0,0:COLOR 1
35 PI=4*ATN(1)
40 FOR HOEK=0 TO 2*PI STEP 0.05
50 PLOT 160+COS(HOEK)*40,90+SIN(HOEK)*40
60 NEXT HOEK
70 GOTO 70
```

In regel 35 wordt de waarde van PI uitgerekend.
Regel 40 laat de hoek die het 'touwtje' maakt niet van 0 tot 360 graden lopen, maar van 0 tot 2*PI. De stapgrootte is aangepast, omdat de punten anders te ver van elkaar vandaan komen te staan.
Regel 50 doet het tekenwerk. Ook dit programma loopt sneller door geen punten te zetten, maar lijnen. Deze bekorten de werking van het programma

natuurlijk alleen als we de stapgrootte gelijk wat vergroten. De cirkel wordt dan iets minder fijn.
Verander in het programma de volgende regels:

```
40 FOR HOEK=0 TO 2*PI STEP 0.2
50 DRAWTO 160+COS(HOEK)*40,90+SIN(HOEK)*40
```

en voeg toe:

```
37 PLOT 160+40,90+0
```

Regel 37 kan zo eenvoudig zijn doordat de cosinus van 0 graden of 0 radialen precies gelijk is aan 1. De sinus van deze hoek is precies 0.

Door in principe uit te gaan van cirkels, maar het gebruik van de sinussen en cosinussen een beetje aan te passen, kan een leuk programma gemaakt worden. Het volgende programma tekent alle lijnen, en laat het punt waar de lijn naar toe getrokken moet worden, over de cirkelboog verspringen. Daardoor ontstaat een verzameling lijnen, die eerst een wirwar lijkt, maar waar al snel een patroon in te zien valt.

```
10 PRINT "SPRONGFACTOR 1";:INPUT A
20 PRINT "SPRONGFACTOR 2";:INPUT B
30 PRINT "STAPGROOTTE   ";:INPUT VE
40 GRAPHICS 8
50 COLOR 1
60 PI=4*ATN(1)
70 PLOT 160,80
80 FOR G=-2*PI TO 2*PI STEP VE
90 CT=COS(A*G)
100 ST=SIN(B*G)
110 DRAWTO 160+70*CT,80+70*ST
120 DRAWTO 160-(70*CT),80-(70*ST)
130 NEXT G
```

Als u achtereenvolgens een 1, een 1 en .1 invoert krijgt u een bol wol te zien.
De combinatie 9, 1, .3 geeft een figuur met bijna rechte zijkanten.
De combinatie 6, 3, .2 geeft een draadjesfiguur.
De combinatie 9, 9, .1 zorgt voor een mooie sneeuwvlok Niet nader te omschrijven, maar fraaie figuren krijgt u door 2, 3, .5 en 4, 4, .5.

Als laatste voorbeeld van het gebruik van cirkels volgt hierna een programma dat op willekeurige plaatsen op het scherm, willekeurige delen van cirkels tekent. Om meer kleur te krijgen is dit programma geschreven voor scherm 11. U kunt het eventueel zelf aanpassen.

```
10 GRAPHICS 11
20 TRAP 120
30 X=INT(RND(0)*80):Y=INT(RND(0)*192)
40 A=RND(0)*2*ATN(1):B=RND(0)*12*ATN(1)
50 KL=INT(RND(0)*17):STRAAL=INT(RND(0)*10)
60 IF A>B THEN Q=B:B=A:A=Q
70 COLOR KL
80 PLOT X+COS(A)*STRAAL,Y+SIN(A)*STRAAL*3.5
90 FOR Z=A TO B STEP 0.4
100 DRAWTO X+COS(Z)*STRAAL,Y+SIN(Z)*STRAAL*3.5
110 NEXT Z
120 GOTO 20
```

Inkleuren

Het is mogelijk om de Atari hele vlakken te laten inkleuren. Dit gaat met de XIO opdracht. Een snelle, maar ingewikkelde opdracht.

```
10 GRAPHICS 7+16
20 FOR KL=1 TO 3
30 COLOR KL
40 XLB=INT(RND(0)*80)
50 YLB=INT(RND(0)*48)
60 XRO=INT(RND(0)*80)+XLB
70 YRO=INT(RND(0)*48)+YLB
80 PLOT XRO,YRO
90 DRAWTO XRO,YLB
100 DRAWTO XLB,YLB
110 POSITION XLB,YRO
120 POKE 765,KL
130 XIO 18,#6,0,0,"S:"
140 NEXT KL
150 FOR WA=1 TO 900:NEXT WA
160 GOTO 10
```

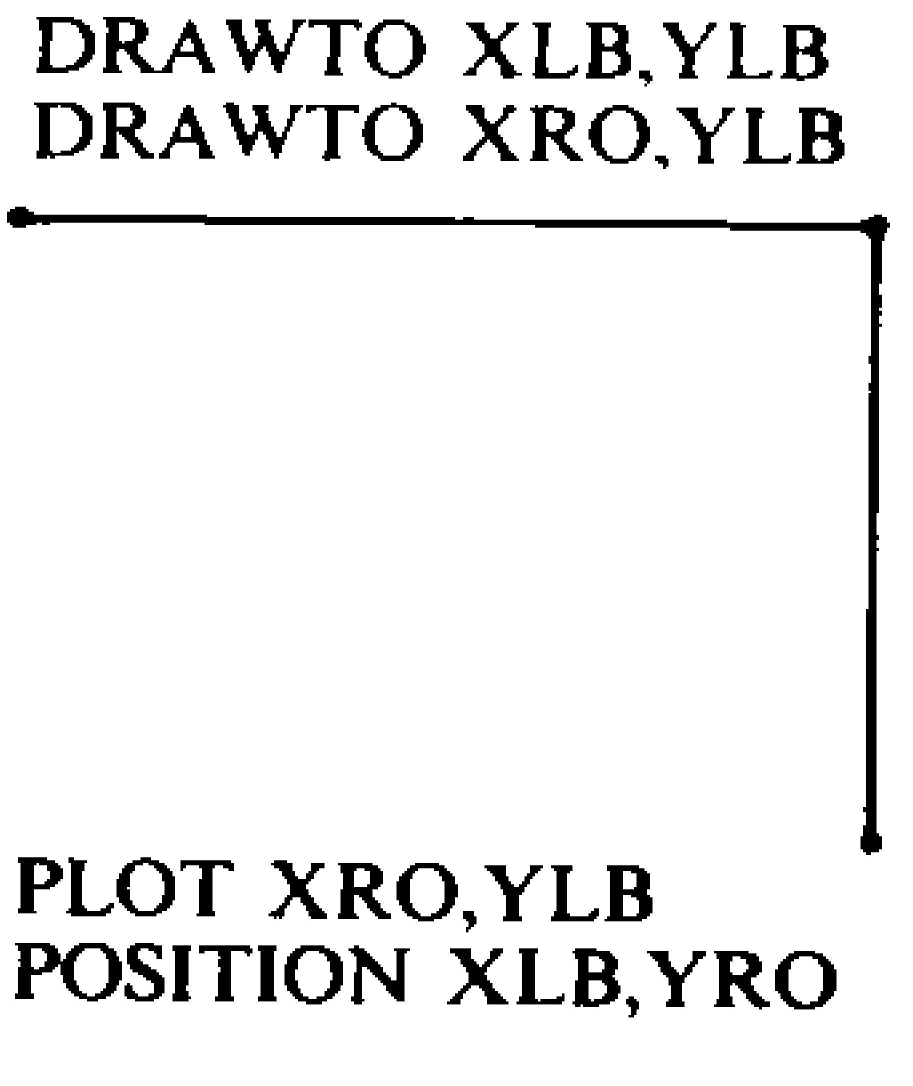
Dit programma laat snel de mogelijkheden, maar ook de gebreken van de XIO opdracht zien. Er worden telkens drie verschillende rechthoeken getekend. Elke rechthoek wordt ingekleurd met een andere kleur. Zodra de rechthoeken ergens overlappen, houdt het inkleuren op, zelfs als de rechthoek aan de andere kant van de overlappende rechthoek verder loopt.
De XIO opdracht ziet er altijd zo uit:

XIO 18, #6, 0, 0, "S:"

U herkent deze opdracht waarschijnlijk al gedeeltelijk van andere IOCB opdrachten. De opdracht werkt via IOCB kanaal #6, het kanaal dat gereser-

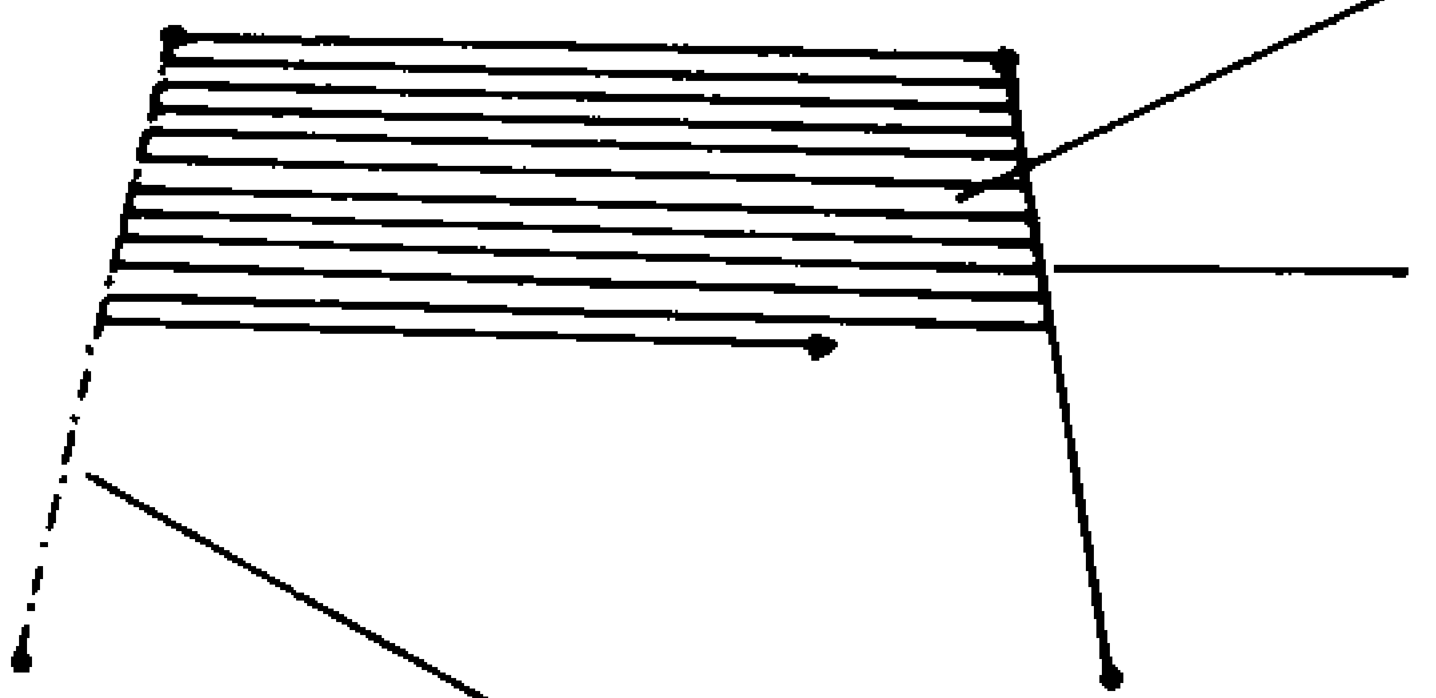
veerd is voor grafische bewerkingen. De S: staat voor uitvoer naar het scherm (Screen). De 18 vertelt de computer dat het om inkleuren gaat. De twee nullen hebben voor deze opdracht verder geen betekenis, maar moeten wel geplaatst worden.

Welk gebied kleurt de XIO opdracht precies?
U moet voor beantwoording van deze vraag eerst letten op het laatste punt waar een grafische opdracht iets gedaan heeft, met andere woorden: de huidige positie van de grafische cursor op het moment dat de XIO opdracht uitgevoerd gaat worden. U weet dat dit punt bepaald wordt door de laatste DRAWTO opdracht (het einde van de lijn) of de laatste PLOT opdracht (het opgegeven punt). Dit punt, de huidige positie van de grafische cursor, is voor de XIO opdracht het beginpunt.
Bij een XIO opdracht hoort altijd een POSITION opdracht. Deze opdracht gebruikte u tot nu toe telkens om op een tekstscherm de positie van een af te drukken tekst te bepalen. Bij de XIO opdracht geeft de POSITION opdracht het eindpunt van de XIO opdracht aan. De coordinaten die achter de POSITION opdracht geplaatst worden, passen in hetzelfde rooster als de de andere coordinaten van het programma.
De XIO opdracht begint met een lijn te trekken van het beginpunt naar het eindpunt. Vervolgens worden er telkens horizontale invullijnen getrokken. Elke invullijn begint bij de net getrokken lijn en loopt naar rechts. Zodra deze invullijn een al ingekleurd punt tegen komt, stopt hij. Dan wordt de volgende horizontale invullijn getrokken. Deze begint een invullijn verder langs de tussen begin- en eindpunt getrokken lijn, en loopt weer van links naar rechts, totdat er een al ingekleurd punt wordt ontmoet.
In bovenstaand programma werden telkens rechthoeken ingekleurd. Eerst plaatst het programma de grafische cursor op de rechter onderhoek. Vandaar wordt een lijn naar de rechter bovenhoek en door naar de linker bovenhoek getrokken. Dat is de laatste grafische opdracht. Het einde van die lijn (de linker bovenhoek) is het beginpunt voor de XIO opdracht.



Met POSITION wordt de linker onderhoek aangegeven. Dit is het eindpunt voor de XIO opdracht. Deze opdracht trekt nu een lijn in de normale COLOR opdracht tussen het beginpunt en het eindpunt van de XIO opdracht.

Vervolgens worden van beginpunt tot eindpunt telkens horizontale lijnen getrokken van links naar rechts. Elke invullijn begint bij de lijn tussen begin- en eindpunt en loopt door tot er rechs ergens een ingekleurde stip of lijn ontmoet wordt.



Uit deze werking van de XIO opdracht volgen gelijk de beperkingen:

- De linkerkant moet een rechte lijn zijn (het hoeft niet perse een verticale lijn te zijn)
- De rechterkant wordt bepaald door ingekleurde punten. Deze kunnen elke vorm aannemen. Is de invullijn gestopt voor een ingekleurde punt, dan wordt het inkleuren op dezelfde lijn niet meer voortgezet.

Het volgende programma laat zien dat de linkerkant van het ingekleurde vlak niet vertikaal hoeft te zijn, en dat de rechterkant ook grillige vormen kan aannemen.

```
10 GRAPHICS 8+16
20 SETCOLOR 2,6,0
30 SETCOLOR 1,6,10:COLOR 1
40 PLOT 250,170
50 FOR GE=1 TO 8
60 READ X,Y
70 DRAWTO X,Y
80 NEXT GE
90 DRAWTO 50,10:REM BEGINPUNT XIO
100 POSITION 120,150:REM EINDPUNT XIO
110 POKE 765,1
120 XIO 18,#6,0,0,"S:"
130 GOTO 130
500 DATA 240,160,290,140,150,130,310
505 DATA 100,280,90,150,85,150,45,250,25
```

Het probleem blijft dat de linkerkant van het in te vullen vlak een rechte lijn moet zijn. Gelukkig heeft u al geleerd dat een kromme lijn niets anders is dan

een verzameling korte rechte lijnen. Dit kan ook gebruikt worden bij de XIO opdracht.

```
10 GRAPHICS 8+16
20 SETCOLOR 2,13,0
30 SETCOLOR 1,13,10:COLOR 1
40 FOR CI=75 TO 225 STEP 75
45 PLOT CI,100
50 DRAWTO CI+50*COS(CI/100),100+50*SIN(CI/100)
60 FOR HOEK=CI/100 TO 6.28 STEP 0.2
70 DRAWTO CI+50*COS(HOEK),100+50*SIN(HOEK)
80 NEXT HOEK
90 DRAWTO CI,100
100 NEXT CI
110 POKE 765,1
120 PLOT 75,100
130 DRAWTO 75+50*COS(0.75),100+50*SIN(0.75)
140 FOR HOEK=0.75 TO 6.28 STEP 0.2
150 DRAWTO 75+50*COS(HOEK),100+50*SIN(HOEK)
160 IF  NOT (HOEK>1.57 AND HOEK<4.71) THEN 170
165 POSITION 75+50*COS(HOEK+0.2),100+50*SIN(HOEK+0.2)
167 XIO 18,#6,0,0,"S:"
170 NEXT HOEK
180 GOTO 180
```

Zoals u ziet wordt op deze wijze de cirkelboog aan de linkerkant keurig ingevuld.

19. PLAYER-MISSILE GRAFIEK

De player-missile grafiek van de Atari is een grafieksoort, die in een apart deel van het geheugen wordt opgeslagen. Door deze aparte opslag is het mogelijk zeer snel en flexibel met deze grafiek te manipuleren. Kortheidshalve zullen vanaf hier player-missile afkorten tot **pm**. De naam is afkomstig van de allereerste arcadespelsystemen van Atari. De bedoeling was dat er een snel te verplaatsen figuur was (de player = speler) vanwaar er een object (de missile - projectiel) kon worden weggeschoten. Deze oorspronkelijke opzet is bij de microcomputers van Atari op vele plaatsen nog verbeterd, de naam is gebleven.

Een tekening in de pm grafiek kan vergeleken worden met een zelf-maak karakter. De pm grafiek is echter een stuk groter, en kan sneller en beter verplaatst worden op het scherm.

De pm grafiek staat helemaal los van de normale tot nu toe behandelde grafiek. Sterker nog: het is mogelijk allebei de soorten grafiek tegelijk op het scherm te hebben, waarbij ze allebei los van elkaar te besturen zijn. Op deze wijze is het mogelijk meer dan de normaal mogelijke kleuren op het scherm te krijgen. U kunt met pm grafiek 5 verschillend gekleurde figuren maken. Deze kunnen ook op het tekstscherm 0 geplaatst worden, zodat ook het tekstscherm kleur krijgt.

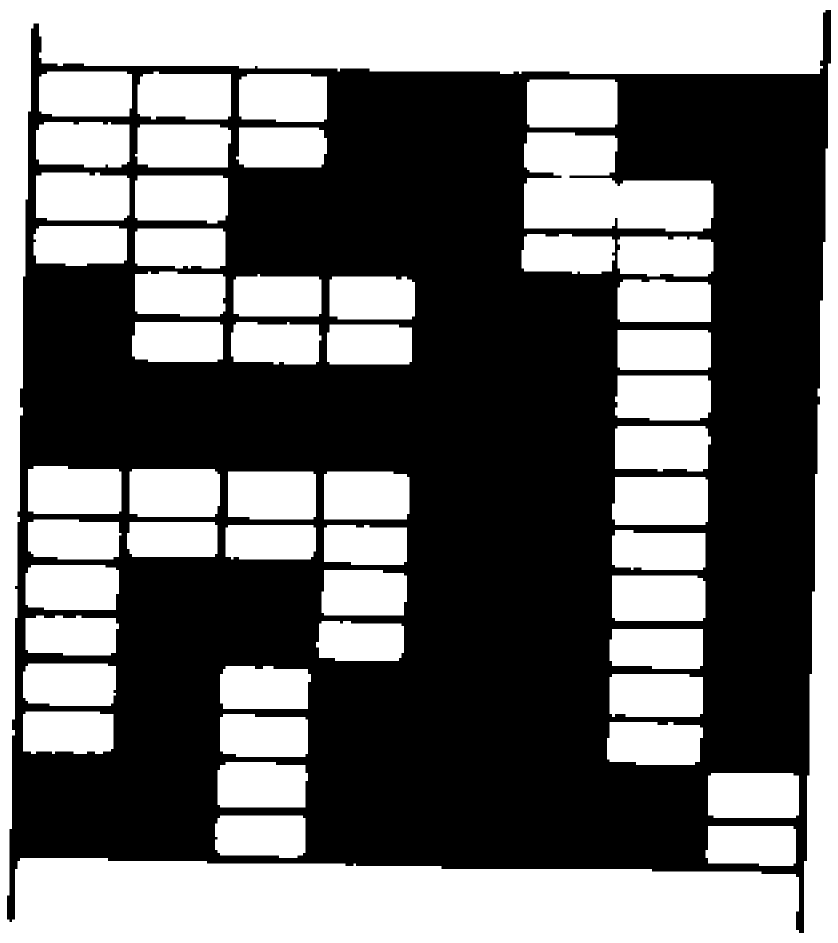
De pm grafiek wordt anders gemaakt dan de normale grafiek. Er komt geen enkele BASIC opdracht bij te kijken. Dit is dan ook een van de grote nadelen van de pm grafiek. Om ermee te kunnen werken, heeft u een heel stel PEEK's en POKE's nodig, laat u zich daardoor niet afschrikken. Het is wat extra werk, en een boel gepuzzel, maar het resultaat is ernaar.

Het maken van een speler

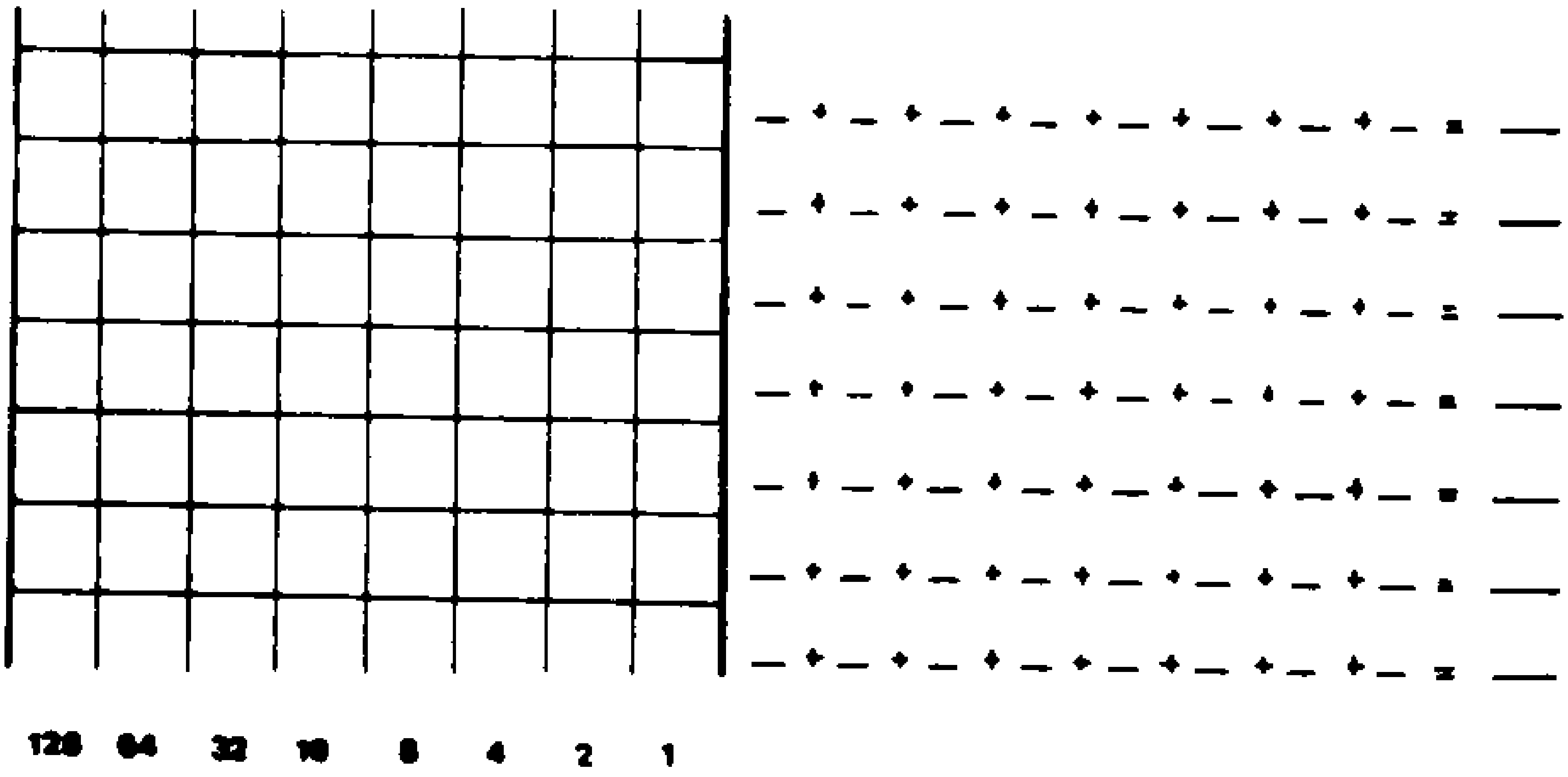
Een object in pm grafiek is een figuurtje, dat als binaire gegevens ergens in het geheugen wordt opslagen. Onafhankelijk van de opslagruimte voor de normale grafische voorstellingen, zodat de pm grafiek onafhankelijk van de normale grafiek kan worden gemanipuleerd.

Zo'n speler, een player, wordt op vergelijkbare wijze als een zelf-maak karakter opgebouwd. Een speler is acht hokjes breed. Dit past precies in een byte geheugenruimte. De hoogte van een speler is anders dan een zelf-maak karakter. Een speler kan maximaal 128 of 255 hokjes hoog zijn. Sterker nog, een speler moet precies die hoogte hebben. Een speler van 128 hokjes (en dus bytes) hoog, gebruikt een byte voor elke twee horizontale lijnen op het scherm. Een speler van 256 bytes hoog, gebruikt een byte voor elke horizontale lijn. Het oplossend vermogen in de verticale richting is voor de 128 bytes player twee keer zo grof als die van de 256 bytes player. Een speler loopt altijd van helemaal bovenaan het scherm tot helemaal onderaan het scherm. Het is echter niet nodig om het gewenste figuurtje zo lang te maken. Door de hele speler eerst met nullen te vullen, en vervolgens alleen de gewenste hokjes in te kleuren, krijgt u een korter figuurtje.

Een speler heeft als eigenschap dat overal waar een hokje (een player is acht hokjes breed en 128 of 256 hokje hoog) aangegeven wordt met een 0, de speler doorzichtig is. Dat wil zeggen dat de onderliggende normale grafiek zichtbaar wordt. Elk hokje dat met een 1 aangegeven wordt, wordt op het scherm ingekleurd. Vergelijk dit met het maken van zelf-maak karakters.
Als voorbeeld dit aapje:



Het is de bedoeling dat elke rij van de gemaakte tekening apart bekeken wordt. Elk gekleurd vakje moet u zien als een 1 en elk niet gekleurd vlakje als een 0. Het binaire getal dat zo doende ontstaat is, het gegeven dat in de byte voor die rij opgeslagen wordt. U kunt het uzelf gemakkelijk maken. U hoeft alleen maar naar de 'waarde' van elk vlakje te kijken. Zo is het hokje helemaal links 128 waard. Het hokje onmiddellijk rechts daarvan is 64 waard. Het hokje daar weer rechts van is 32 waard, enzovoorts tot het meest rechtse hokje dat slechts 1 waard is. Door per rij de waarden van alle gekleurde vlakken bij elkaar op te tellen, krijgt u de totale waarde van elke rij in decimale notering. U kunt daarvoor het onderstaande rooster gebruiken:

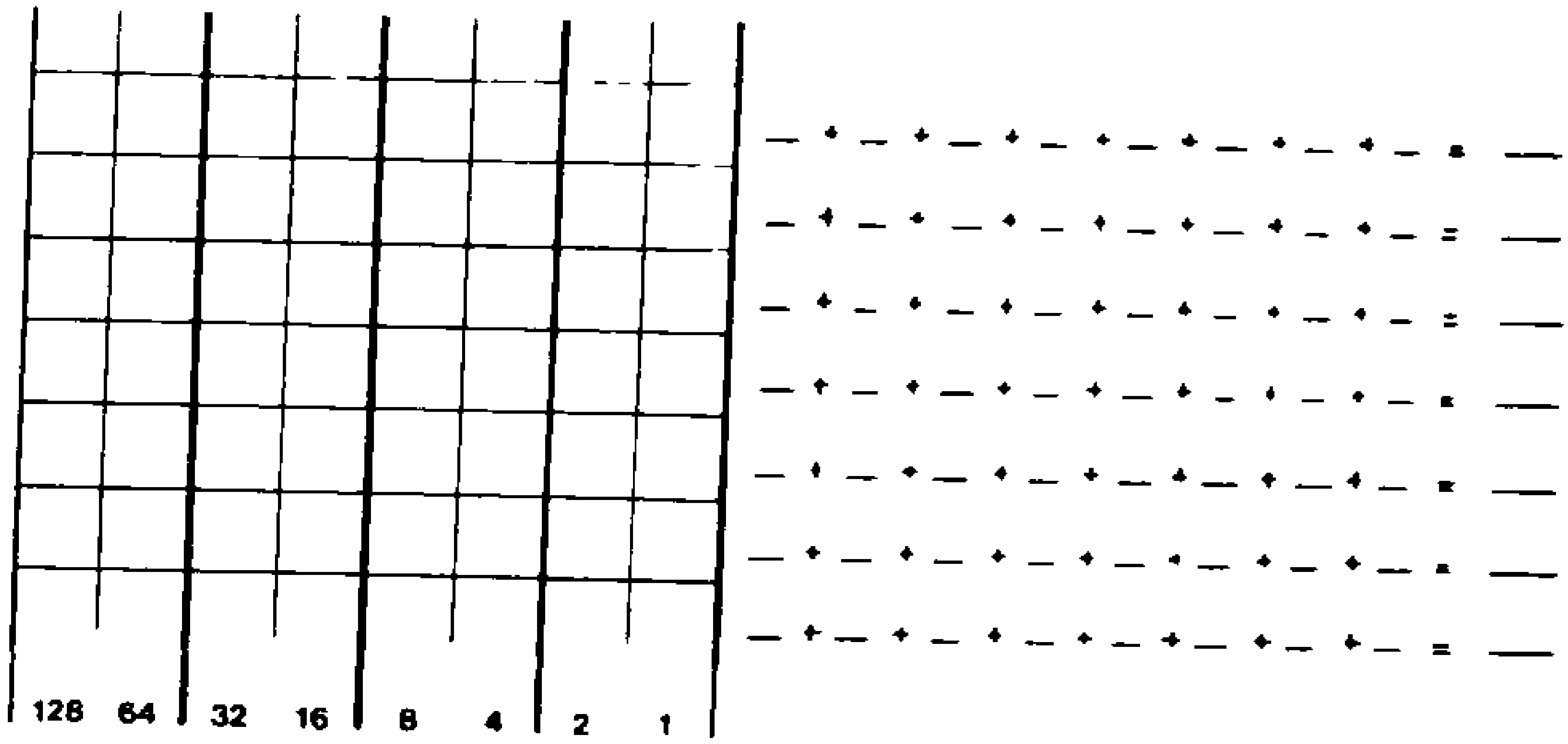


Als voorbeeld bekijken we de bovenste rij van het aapje. Het meest linkse hokje is niet gekleurd: geen 128. Het volgende hokje is ook niet gekleurd: geen

64. Het daarop volgende hokje is ook niet gekleurd: geen 32. Het vierde hokje is echter wel gekleurd: 16. Het volgende hokje is ook weer gekleurd: 8. Het hokje rechts daarvan is leeg: geen 4. Het op een na laatste hokje is gekleurd: 2. En het laatste hokje is ook gekleurd: 1. Tellen we deze waarden bij elkaar op dan krijgen we **16+8+2+1=27.** De decimale notering voor het binaire getal is **00011011.** Op dezelfde wijze kunnen alle rijen van de speler uitgerekend worden.

Het maken van een projectiel

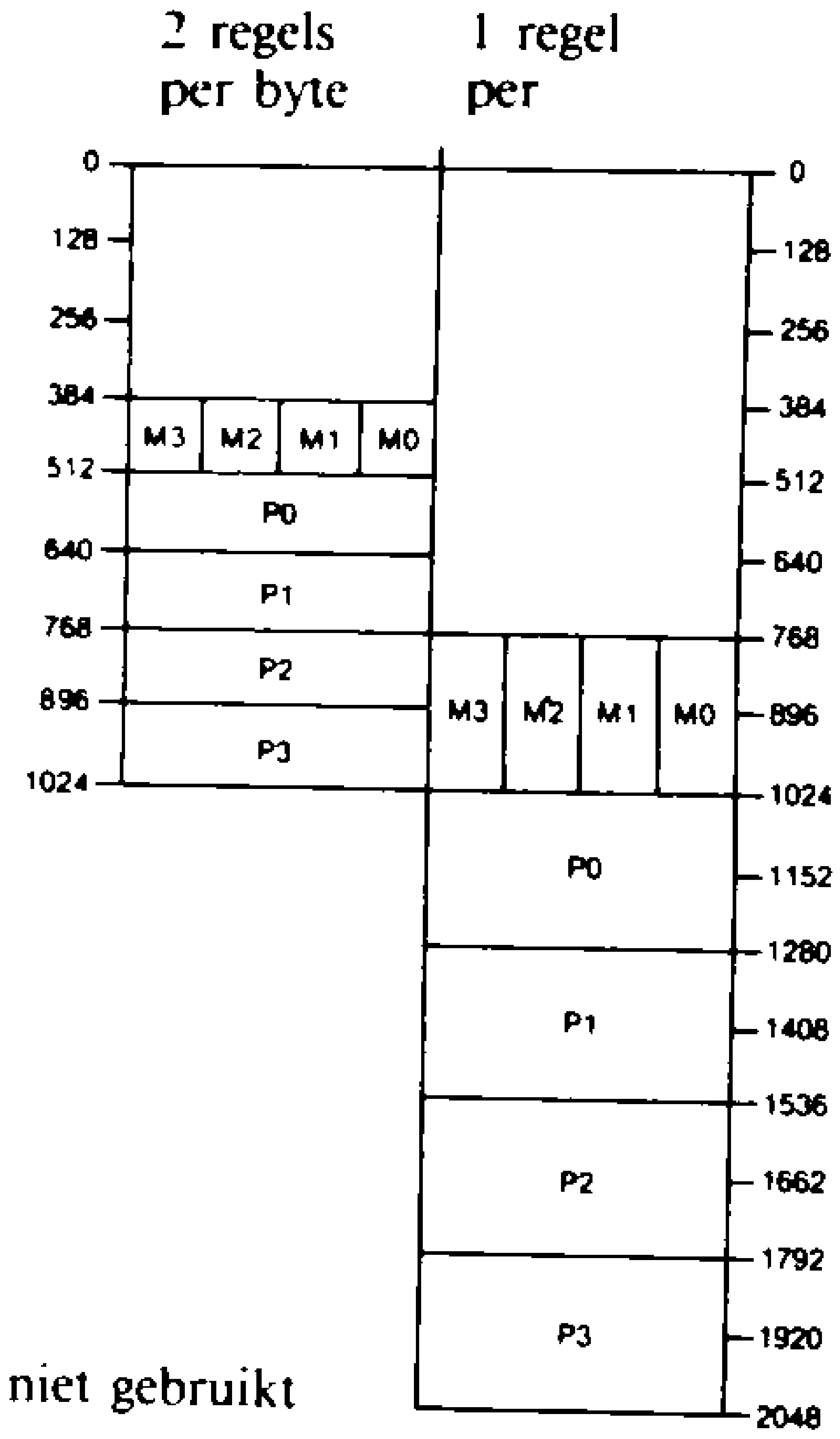
Een projectiel of een missile, wordt op vergelijkbare wijze gemaakt als een speler. Er is eigenlijk nauwelijks verschil tussen die twee. Alleen zijn projectielen slechts twee hokjes breed. Er gaan dus vier projectielen op de plaats van een enkele speler. De vier projectielen zijn ook naast elkaar in het geheugen geplaatst. Dat betekent dat we weer gebruik kunnen maken van een rooster van acht hokjes breed. De twee meest linkse hokjes zijn voor projectiel 3. De volgende twee hokjes zijn voor projectiel 2. De daarop volgende twee hokjes zijn voor projectiel 1. De twee meest rechtse hokjes zijn voor projectiel 0.



Met slechts twee hokjes naast elkaar, is het niet zo goed mogelijk erg spannende projectielen te maken. U kunt dat bezwaar opheffen door twee projectielen achter elkaar te zetten, en te zorgen dat deze gedurende het programma achter elkaar blijven staan.

Geheugengebruik

Door spelers van 256 bytes hoog te maken, heeft u twee maal zoveel geheugenruimte nodig. Voor een stel van vier spelers en vier projectielen van elk 128 bytes heeft u 1024 bytes (=1 kilobyte) geheugenruimte nodig. De verdeling van de nodige geheugenruimte is als volgt:



De benodigde geheugenruimte (1024 of 2048 bytes) moet gereserveerd worden. Het gereserveerde stuk geheugen moet beschermd worden tegen BASIC, tegen het bedrijfssysteem en tegen het gebruik van het geheugen voor de normale grafiek.

De gereserveerde geheugenruimte is 1 kbyte groot en begint op een 1 kbyte grens, of is 2 kbytes groot en begint op een 2 kbyte grens.

Zoals al gezegd, bestaan voor de pm grafiek geen BASIC opdrachten. Het grootste probleem bij het werken met de pm grafiek is de wijze waarop het geheugen ingedeeld moet worden. Omdat dit een grondige kennis van de interne werking van de Atari vraagt, voert het te ver om dit alles in dit boek te behandelen. Om te zorgen dat u toch de meeste mogelijkheden van de pm grafiek kunt benutten vindt u in dit hoofdstuk een uitleg van alle handelingen die verricht moeten worden om de pm grafiek op het scherm te zetten. Waar en hoe de pm grafiek in het geheugen geplaatst wordt, is minder interessant. U kunt daarvoor gebruik maken van de in dit hoofdstuk gegeven routines die u zelf in uw programma's kunt inbouwen.

Het volgende programma stelt de pm grafiek in op een tekstscherm 0. De vertikale resolutie is een byte per twee regels. Als voorbeeld van een speler wordt een giraf getekend.

```
10 GRAPHICS 0
15 SETCOLOR 2,0,0:FOR Q=1 TO 22:PRINT Q:NEXT Q
20 GOSUB 5000
30 X=150:Y=20:REM POSITIE OP SCHERM
40 FOR I=SPELER0+Y TO SPELER0+Y+26
50 READ GE:POKE I,GE
60 NEXT I
70 POKE 53248,X
80 END
500 DATA 2,2,7,7,4,4,4,4,4,4,4,4,12,12,62,62
505 DATA 126,126,126,126,38,38,98,98,66,66,66,66
5000 A=PEEK(106)-8
5010 POKE 54279,A
5020 PMBEGIN=256*A
5030 POKE 559,46
5040 POKE 53277,3
5050 POKE 704,213
5060 SPELER0=PMBEGIN+512
5070 FOR I=SPELER0 TO SPELER0+128:POKE I,0
5075 NEXT I
5080 RETURN
```

Als u dit programma intikt en runt ziet u rechts op het beeld een stel getallen verschijnen en vervolgens midden op het beeld een gele giraf. De cijfers zijn gemaakt door de normale grafische en tekstopdrachten (regel 15) en de giraf is gemaakt met pm grafiek.
Om uzelf ervan te overtuigen dat deze twee methodes van weergave op het beeldscherm niets met elkaar te maken hebben, kunt u een direkte LIST

opdracht geven. De listing rolt over het scherm, maar de giraf blijft rustig staan. Merkt u ook op dat ondanks het feit dat in schermmodus 0 maar een kleur gebruikt kan worden, de giraf toch een derde kleur heeft?

In regel 20 wordt naar de subroutine gesprongen die de pm grafiek instelt. In regel 30 wordt de positie van de speler bepaald. De variabele X bepaalt de horizontale positie van de speler. De variabele Y bepaalt de vertikale positie. Vooral het veranderen van de horizontale positie van een speler is een uiterst eenvoudige en razendsnelle aangelegenheid. De horizontale positie van elke speler wordt in een speciaal geheugenadres vastgelegd. Dat zijn voor de vier spelers en de projectielen:

Geheugenadres	Horizontale positie van
53248	speler 0
53249	speler 1
53250	speler 2
53251	speler 3
53252	projectiel 0
53253	projectiel 1
53254	projectiel 2
53255	projectiel 3

Door in het juiste geheugenadres telkens een andere waarde te POKEn, kan de speler heel snel en vloeiend over het scherm verplaatst worden. Opvallend daarbij is dat u als programmeur niet hoeft te letten op het uitwissen van de 'oude' speler. De speler wordt namelijk niet opnieuw getekend en de oude speler daarna gewist, maar de plaats van de speler wordt veranderd. Dit gaat vele malen sneller.
Verandert u regel 70 van het programma in:

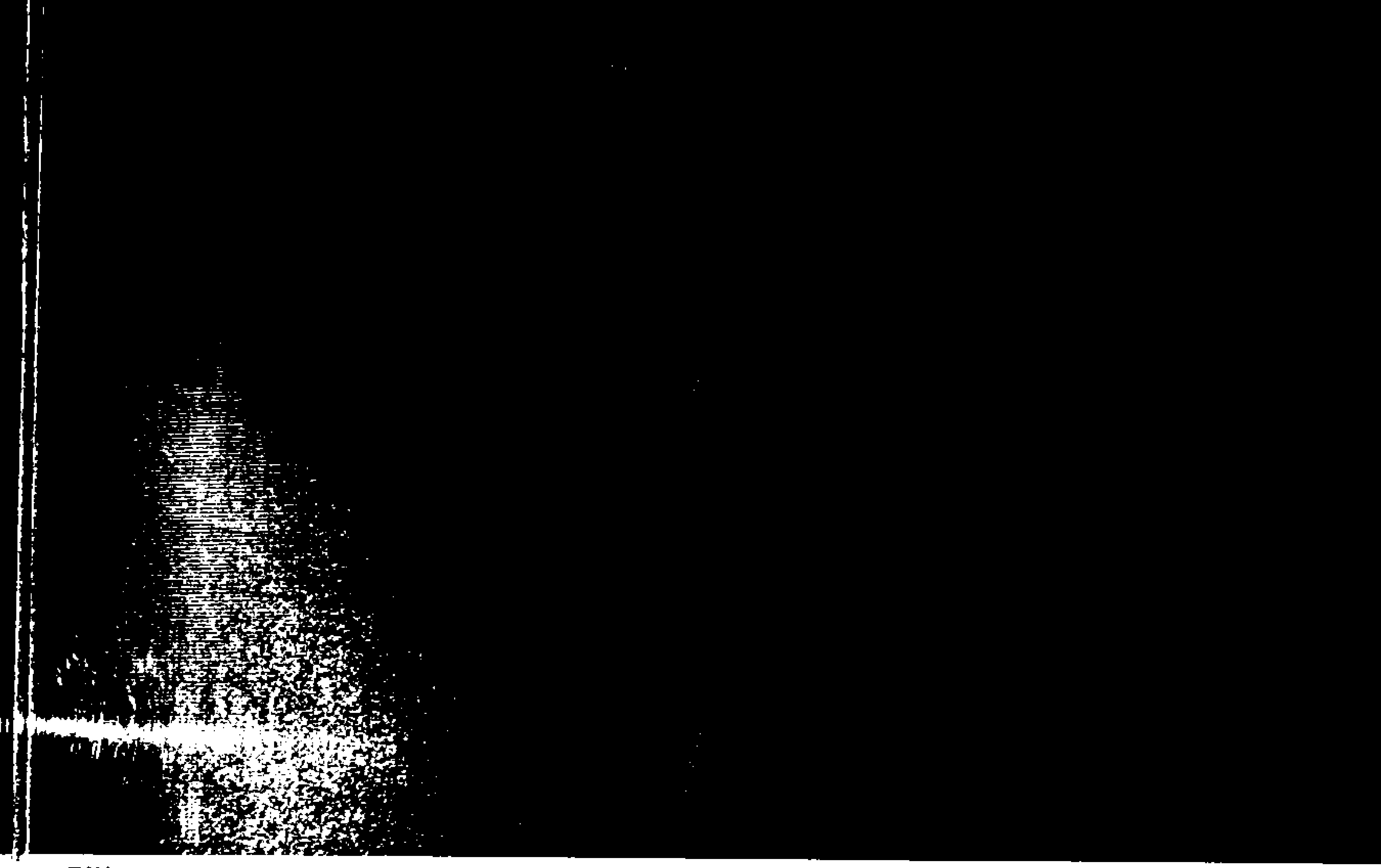
```
70 FOR X=0 TO 227 : POKE 53248,X : NEXT X
```

Run het programma nu opnieuw en zie hoe snel de giraf over het scherm holt.

Het bereik van X in de nieuwe regel 70 is tevens het horizontale bereik van de pm grafiek. U kunt een speler niet verder naar links plaatsen dan positie 0. Een speler kan niet verder naar rechts dan positie 227.
Deze posities zijn niet direkt gerelateerd aan het beeldscherm. Positie 0 is altijd links van het beeldscherm. Plaatst u de speler op positie 0, dan is deze niet op het scherm zichtbaar. Probeert u dat door regel 70 weer te veranderen:

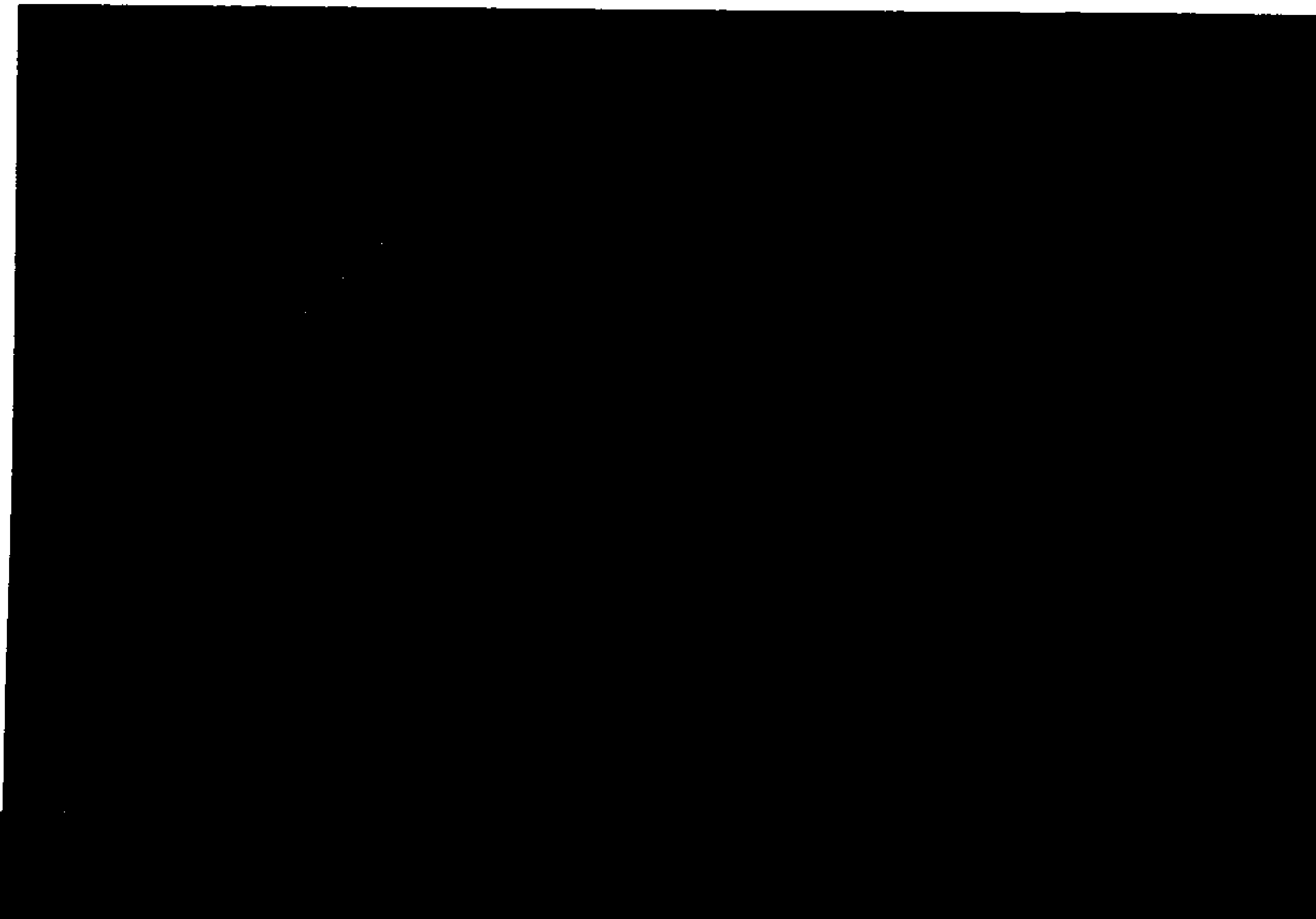
```
70 FOR X=227 TO 0 STEP -1 : POKE 53248,X : NEXT X
```

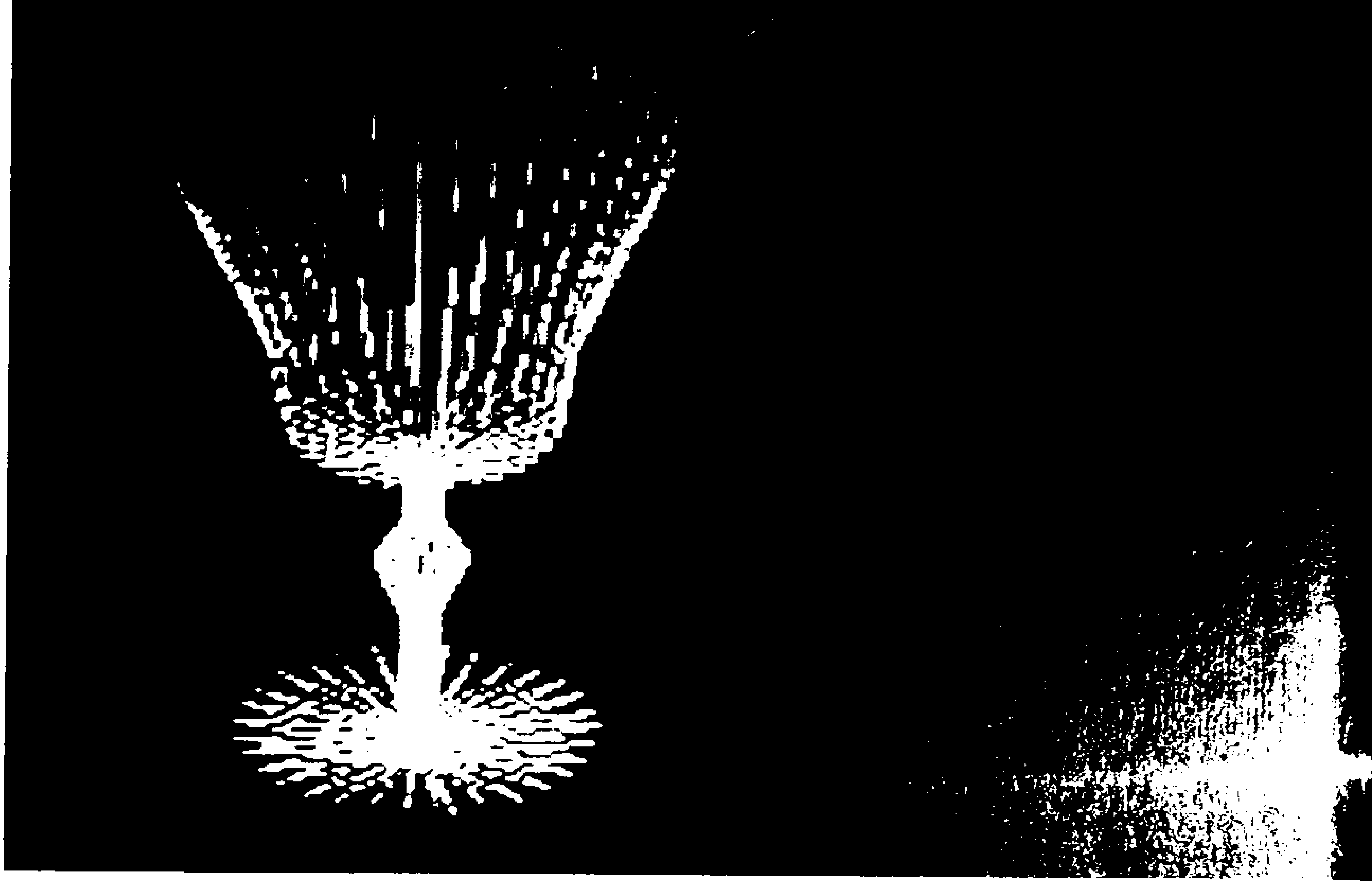
De giraf loopt nu achteruit en verdwijnt aan de linkerkant van het scherm. Vanaf welke positie de speler rechts niet meer op het scherm te zien is, is afhankelijk van de breedte van de speler. Deze is in te stellen als enkele



Cilinders programma *blz. 160*

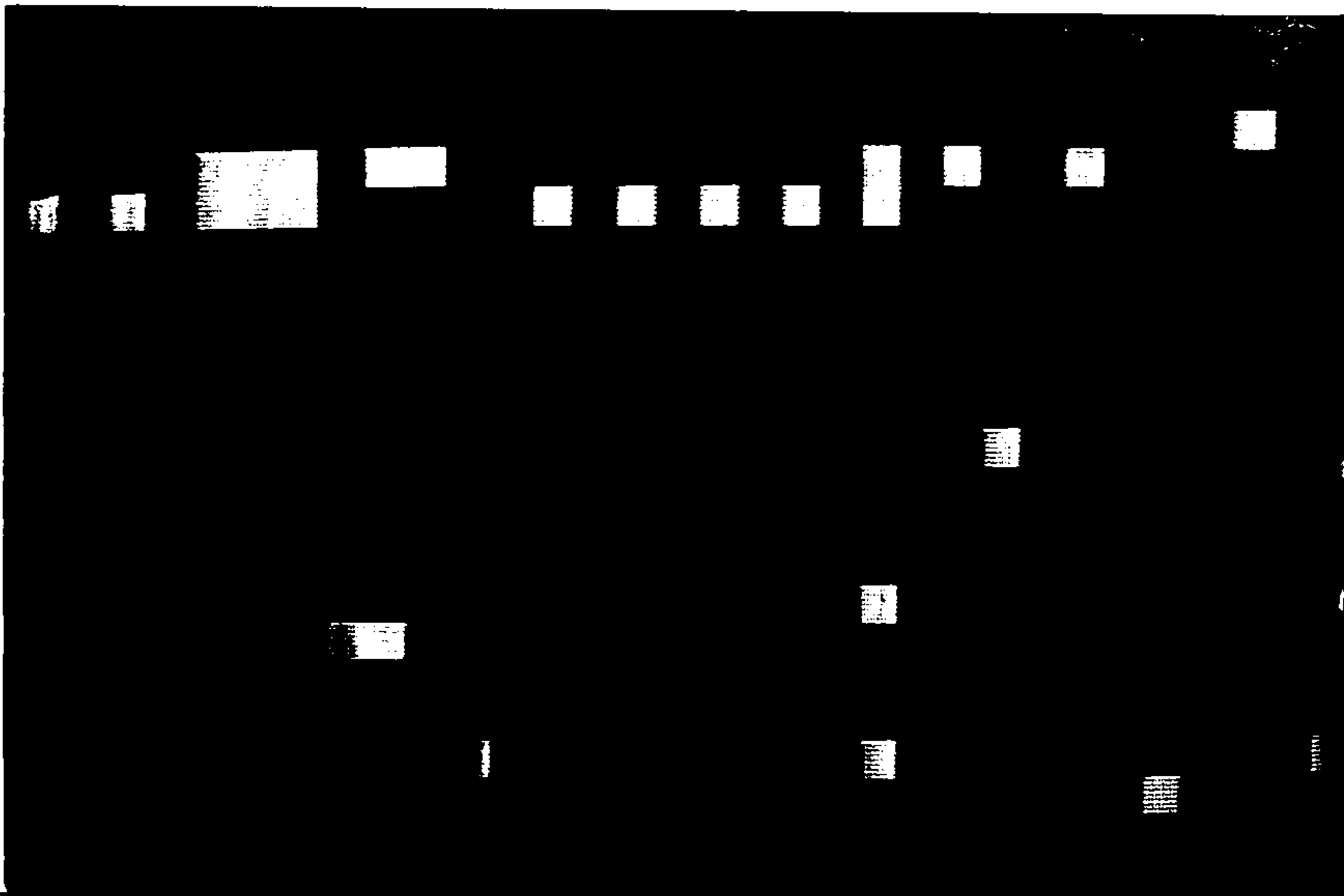
Kubus met dieptesuggestie door schaduwwerking programma *blz. 248*





Wijnglas programma zie *blz.* 262

Gekleurde blokjes programma onderbroken *blz.* 188



breedte, dubbele breedte en vier-dubbele breedte. Meer hierover later in dit hoofdstuk.
U kunt de horizontale positie ook door een direkte opdracht beïnvloeden. Als het programma afgelopen is en de giraf staat niet meer op het scherm kunt u door direkt **POKE 53248,100** in te tikken, de giraf weer op het scherm laten verschijnen.

Regel 500 van het programma geeft de data die samen de giraf vormen.

Instellen van pm grafiek

De subroutine van regel 5000 tot en met 5080 in het vorige programma zorgt voor het instellen van de pm grafiek. Om de pm grafiek in te stellen is een serie PEEKs en POKEs nodig. Het voert te ver om alle achtergronden van elke PEEK en POKE te behandelen, maar hieronder vindt u de hoofdlijnen, zodat u zelf naar wens de routine kunt aanpassen.
Regel 5000 PEEKt de waarde van geheugenlokatie 106 en trekt daar 8 vanaf. Hierdoor wordt bepaald waar de pm grafiek in het geheugen zal worden geplaatst. In plaats van 8 kunt u ook een kleiner getal nemen, maar u loopt dan de kans dat uw gewone grafiek (of tekst) door de pm grafiek wordt vernietigd. Als u een meer uitgebreide PM grafiek wilt (een resolutie waarbij elke byte slechts een enkele regel bestuurt) dan zult u een grotere waarde in plaats van de 8 moeten nemen. Hetzelfde geldt voor het kiezen van de schermmodus. In het algemeen geldt dat hoe meer geheugenruimte de schermmodus vraagt, hoe groter het getal in plaats van de 8 moet zijn. Zie ook de volgende programma's.
Regel 5010 plaatst de waarde van A in geheugenlokatie 54279. Daardoor weet de computer waar de pm grafiek is opgeslagen.
Regel 5020 rekent aan de hand van de waarde van A, de werkelijke plaats van het begin van de pm grafiek in het geheugen, uit. Dat A daarvoor met 256 vermenigvuldigd moet worden, heeft te maken met het feit dat in één enkele byte geheugenruimte geen getallen groter dan 255 passen.
De werkelijke beginplaats van de pm grafiek in het geheugen, noemen we PMBEGIN. Het is dit adres van waaruit alle andere adressen, die te maken hebben met de pm grafiek, berekend worden. Zie hiervoor ook de tabel met het gebruik van geheugenruimte van de pm grafiek.
Regel 5030 POKEt in geheugenlokatie 559 de waarde 46. Hiermee wordt aan de computer opgegeven dat de gebruikte pm grafiek dubbele lijn resolutie gebruikt (een byte stuurt telkens twee regels van de speler aan). Om aan te geven dat u enkele lijn resolutie gaat gebruiken, POKEt u in deze lokatie (559) de waarde 62.
Regel 5040 zet de pm grafiek aan door in lokatie 53277 de waarde 3 te POKEn. Als u hier een 0 POKEt, wordt de pm grafiek buiten werking gesteld. Probeer dat uit door de volgende twee direkte POKE opdrachten:

POKE 53277,0	(de giraf verdwijnt)
POKE 53277,3	(de giraf verschijnt weer)

Regel 5050 bepaalt de kleur van speler 0. De waarde die u moet POKEn is gelijk aan de waarde die u zou gebruiken bij het POKEn van een SETCOLOR adres: 16 * kleur + helderheid. Ook dit adres kunt u direkt POKEn waardoor de kleur van de giraf verandert. Probeer de kleur van de giraf te veranderen door verschillende waarden in lokatie 704 te POKEn.
De volgende regel laat alles zien:

```
FOR X=0 TO 255:POKE 704,X:FOR WA=1 TO 50:NEXT WA:NEXTX
```

Regel 5060 bepaalt waar de geheugenlokaties voor speler 0 beginnen. In de tabel die het geheugengebruik voor de pm grafiek weergeven, is af te lezen dat speler 0 van PMBEGIN+512 tot PMBEGIN+640 loopt. Speler 1 loopt van PMBEGIN+640 tot PMBEGIN+768 enz. In de dubbele-lijn resolutie (elke byte zorgt voor twee lijnen), heeft elke speler 128 bytes geheugenruimte nodig.
Regel 5070 maakt het gebied voor speler 0 helemaal leeg. Elke byte wordt gevuld met een 0. Daardoor ontstaat een hoge kolom (128*2 lijnen hoog) van 8 blokje breed (een byte) die helemaal doorzichtig is. Alleen daar waar de blokjes niet aangegeven zijn met een 0, wordt een blokje ingekleurd.
Regel 5080 beeindigt de subroutine.

Door deze routine is de pm grafiek ingesteld en is speler 0 leeg gemaakt. Het is de taak van het hoofdprogramma om speler 0 weer van informatie te voorzien, de speler te plaatsen en van kleur te voorzien.

Enkele lijn resolutie

Het volgende programma geeft u een routine die vergelijkbaar is met de vorige, maar die u één enkele lijn resolutie geeft. Dat wil zeggen dat elke byte slechts een enkele lijn (van 8 hokjes breed) aanstuurt. U kunt daardoor veel nauwkeuriger tekeningen maken.

```
10 GRAPHICS 5
15 SETCOLOR 2,0,0:FOR Q=1 TO 22:PRINT Q:NEXT Q
20 GOSUB 5000
30 X=150:Y=60:REM POSITIE OP SCHERM
40 FOR I=SPELER0+Y TO SPELER0+Y+30
50 READ GE:POKE I,GE
60 NEXT I
70 POKE 53248,X
80 END
500 DATA 2,2,7,7,6,4,4,4,4,4,4,4,4,4,12,30,62,126
505 DATA 126,126,126,126,38,38,98,98,66,66,66,66,34
5000 A=PEEK(106)-16
5010 POKE 54279,A
5020 PMBEGIN=256*A
5030 POKE 559,62
5040 POKE 53277,3
```

```
5050 POKE 704,213
5060 SPELERO=PMBEGIN+1024
5070 FOR I=SPELERO TO SPELERO+256:POKE I,0
5075 NEXT I
5080 RETURN
```

Dit programma tekent dezelfde giraf, maar nu iets nauwkeuriger. Dat is mogelijk, omdat elke lijn nu apart ingekleurd kan worden.
In tegenstelling tot het vorige programma wordt dit keer de giraf op het grafische scherm 5 getekend. U kunt dit benadrukken door op dat grafische scherm iets te tekenen. Voeg de volgende regels toe:

```
22 FOR Q=1 TO 10:K=INT(RND(0)*80):L=INT(RND(0)*40)
24 M=INT(RND(0)*80:N=INT(RND(0)*40)
26 COLOR 1:PLOT K,L:DRAWTO M,N
28 NEXT Q
```

In vergelijking met het vorige programma is regel 40 aangepast. Er zijn nu meer gegevens beschikbaar voor speler 0.

De subroutine voor het instellen van de pm grafiek is op enkele punten afwijkend:
In regel 5000 wordt een veel grotere waarde afgetrokken van de PEEK(106).
Dit gebeurt omdat de enkele-lijn resolutie tweemaal zoveel geheugenruimte nodig heeft, en omdat het grafische scherm 5 meer geheugenruimte nodig heeft als het tekstscherm 0.
In regel 5030 wordt in lokatie 559 de waarde 62 gePOKEt. Deze waarde geeft de computer te kennen dat er gewerkt moet worden met enkele-lijn resolutie.
In regel 5060 wordt het beginadres van speler 0 anders. Zie de tabel met de verdeling van de geheugenruimte voor pm grafiek.
In regel 5070 zijn het begin- en eindadres voor het wissen van het voor de speler bedoelde geheugengebied iets anders. Ook dit is weer af te lezen uit de tabel. Speler 0 heeft nu 256 geheugenbytes nodig, lopend van PMBEGIN+1024 tot PMBEGIN+1024+256.

Dubbelgroot en beweging

Het volgende programma laat zien hoe er meer dan één speler tegelijk gebruikt kan worden, en hoe deze spelers verplaatst kunnen worden. Daarnaast laat het programma zien dat het mogelijk is om de spelers een verschillend formaat te geven.

```
10 GRAPHICS 5+16
20 SETCOLOR 4,0,6:SETCOLOR 0,0,0
30 SETCOLOR 1,0,4:SETCOLOR 2,0,12
40 COLOR 2
50 FOR Y=25 TO 47:PLOT 0,Y:DRAWTO 79,Y:NEXT Y
```

```
60 COLOR 1
70 FOR X=0 TO 60 STEP 10
80 PLOT X,25:Y=INT(RND(0)*12):DRAWTO X,Y
85 DRAWTO X+8,Y:DRAWTO X+8,25
90 NEXT X
100 COLOR 3
105 FOR LA=7 TO 67 STEP 20
110 PLOT LA,28:DRAWTO LA,15:DRAWTO LA-3,12
115 DRAWTO LA+3,12:DRAWTO LA,15:NEXT LA
120 GOSUB 5000:REM PM GRAFIEK
125 REM SPELERS
130 XA=90:YA=63
140 FOR I=SPELER0+YA TO SPELER0+YA+15
150 READ A:POKE I,A:NEXT I
160 XB=150:YB=80
170 FOR I=SPELER1+YB TO SPELER1+YB+15
180 READ B:POKE I,B:NEXT I
190 XC=150+32:YC=80
200 FOR I=SPELER2+YC TO SPELER2+YC+15
210 READ C:POKE I,C:NEXT I
220 POKE 704,118:REM KLEUR SPELER 0
230 POKE 705,56:REM KLEUR SPELER 1
240 POKE 706,56:REM KLEUR SPELER 2
250 POKE 53256,0:REM SPELER 0 NORMAAL
260 POKE 53257,3:POKE 53258,3:REM 1,2=4-DUB
270 REM POSITIES
275 DKL=10
280 XA=XA+0.5:XB=XB-2:XC=XB+32
290 IF XA=219 THEN XA=0
300 IF  NOT XB=0 THEN 310
305 XB=200:KL=KL+DKL:POKE 705,KL:POKE 706,KL
307 IF KL>246 OR KL<6 THEN DKL=-DKL
310 POKE 53248,XA
320 POKE 53250,XC
330 POKE 53249,XB
340 GOTO 280
400 END
499 REM DATA SPELER 0
500 DATA 24,24,16,16,56,56,84,84,58,58
505 DATA 24,24,104,104,140,140
509 REM DATA SPELER 1
510 DATA 0,0,0,0,1,1,2,2,63,63
515 DATA 127,127,231,231,24,24
519 REM DATA SPELER 2
520 DATA 0,0,252,252,84,84,82,82,254,254
525 DATA 254,254,231,231,24,24
5000 A=PEEK(106)-16
```

```
5010 POKE 54279,A
5020 PMBEGIN=256*A
5030 POKE 559,46
5040 POKE 53277,3
5050 SPELER0=PMBEGIN+512
5060 SPELER1=PMBEGIN+640
5070 SPELER2=PMBEGIN+768
5080 FOR I=PMBEGIN+512 TO PMBEGIN+896:POKE I,0
5085 NEXT I
5090 POKE 623,1:REM SPELERS VOORRANG
5100 RETURN
```

In regel 10 wordt een grafisch scherm 5 geopend zonder tekstvenster (+16). In de regels 20 en 30 worden de vier kleuren van deze grafische modus ingesteld, zodat het allemaal grijstinten zijn. Het programma laat een mannetje zien in een grijze straat. Om het effect van een saaie straat te versterken zijn alle kleuren in een grijstint.
Regel 50 zorgt voor het asfalt.
De regels 60-90 zorgt voor een stel huizen of flats.
De regels 100-110 zorgen voor vier lantaarnpalen in wit.
Regel 120 springt naar de subroutine die de pm grafiek instelt.
Regel 130 stelt de horizontale waarde (XA) en de verticale waarde (YA) van speler 0 in. Vervolgens worden in de regels 140 en 150 de waarden die speler 0 zijn vorm moeten geven, ingelezen.
Regel 160 stelt de horizontale waarde (XB) en de verticale waarde (YB) van speler 1 in.
In de regels 170 en 180 worden de waarden gelezen die ervoor zorgen dat speler 1 er als de voorkant van een auto uitziet.
Regel 190 stelt de horizontale (XC) en verticale (YC) van speler 2 in. De regels 200 en 210 lezen de waarden voor speler 2: de achterkant van de auto.
De regels 220-240 stellen de kleuren van de drie spelers in. Omdat speler 1 en speler 2 samen een enkel object (een auto) vormen, hebben deze dezelfde kleur, en moeten zij dat blijven houden gedurende het hele programma.
De regels 250 en 260 laten nieuwe POKE adressen zien:

adres	funktie
53256	formaat speler 0 (0=normaal, 1=dubbel, 3=vier-dubbel)
53257	formaat speler 1 (0=normaal, 1=dubbel, 3=vier-dubbel)
53258	formaat speler 2 (0=normaal, 1=dubbel, 3=vier-dubbel)
53259	formaat speler 3 (0=normaal, 1=dubbel, 3=vier-dubbel)

In het programma heeft de wandelaar een normaal formaat. De auto heeft het vierdubbele formaat. Dat wil zeggen dat de speler (in dit geval, allebei de spelers die de auto vormen) vier keer zo breed weergegeven worden. Dit verandert niets aan het oplossend vermogen. U kunt ook bij dubbele of vierdubbele spelers slechts 8 hokjes in de breedte vast stellen. De hokjes worden al-

leen elk twee keer of vier keer zo breed weergegeven. De hoogte van de speler is voor elk formaat gelijk. U kunt er vaak beter van te voren rekening mee houden of u een speler breder gaat weergeven of niet. Als de auto de normale hoogte had, was het een zeer lang gerekte lage vorm geworden. Door alle DATA opdrachten die de auto vormen, twee keer op te nemen, werd de auto twee keer zo hoog.

De regels 270-300 berekenen waar de spelers op het scherm moeten worden geplaatst. Door de voetganger langzamer te verplaatsen dan de auto, wordt de weergave levensechter.
Regel 300 zorgt er tevens voor dat de kleur van de auto telkens wordt gewisseld als er een nieuwe auto op het scherm gaat komen. Daardoor komt er een bonte stoet van auto's langs.
De regels 310-330 POKEn de in de vorige regels gevonden waarden in de geheugenlokaties die bepalen waar de spelers komen te staan.
Regel 340 maakt de lus rond.
De regels 499-520 bevatten de gegevens die de spelers samenstellen.

De regels 5000-5100 vormen de subroutine die de pm grafiek instellen. U kent dit gedeelte al van de vorige programma's. Let u echter op de volgende punten.
In regel 5000 wordt 16 afgetrokken van de PEEK(106) waarde. U kunt hier volstaan met 12 af te trekken. Omdat niet elke Atari even veel geheugenruimte heeft, kan het voorkomen dat u niet genoeg geheugenruimte heeft voor de pm grafiek. Probeert u dan om een minder geheugen vragende grafisch scherm als achtergrond te gebruiken, en trekt u een kleinere waarde van PEEK(106) af. Als deze waarde te klein wordt, krijgt u ongewenste effecten. De pm grafiek beinvloedt in dat geval de normale grafiek of de BASIC. Probeer in het programma de waarde 16 te vervangen door 8 of een nog kleinere waarde. Zie hoe de normale grafiek verandert in een onbegrijpelijke toestand.
In regel 5030 wordt in geheugenlokatie 559 de waarde 46 gePOKEt. Het gaat om dubbele-lijn resolutie.
In de regels 5050-5070 worden de beginadressen van drie spelers vast gesteld. U heeft maximaal vier spelers tot uw beschikking (met eventueel een samengaan van de projectielen tot een extra speler).
In regel 5080 wordt een groter stuk dan in de vorige programma's gewist door vulling met nullen. Dit komt doordat er nu drie spelers gebruikt worden in plaats van een.
Regel 5090 kent nog een nieuw POKE adres. De inhoud van adres 623 bepaalt welke afbeeldingen op het scherm 'voorrang' hebben. Als een bepaalde afbeelding voorrang heeft boven een andere afbeelding wil dat zeggen dat de voorrang hebbende afbeelding voor de andere afbeelding langs schuift. Als twee afbeeldingen overlappen , blijft de afbeelding met voorrang zichtbaar en verdwijnt de andere afbeelding onder deze voorrang hebbende afbeelding. Een afbeelding in de pm grafiek wordt aangeduid door het spelersnummer. Een afbeelding in de normale grafiek wordt aangeduid door het gebruikte kleurregister.

De volgende waarden kunnen in adres 623 gePOKEt worden:

Adres 623: voorrang	
1	alle pm spelers hebben voorrang over alle gewone grafiek
4	alle gewone grafiek heeft voorrang over pm spelers
2	gemengd: eerst pm speler 0 en 1, dan gewone grafiek en tenslotte pm spelers 2 en 3
8	gemengd: eerst gewone grafiek in de kleur van de registers 0 en 1, dan alle pm spelers en tenslotte de gewone grafiek in de kleur van de registers 2 en 3

U kunt dit uitproberen door de waarde in regel 5090 telkens te veranderen. Verander regel 5090 eerst in:

```
5090 POKE 623,4
```

Als u nu het programma runt zult u alleen het hoofd van de wandelaar zien, en de auto en de wandelaar telkens even zien als zij aan de zijkanten van de tekening verschijnen. Alle gewone grafiek heeft voorrang boven de pm spelers. Verander regel 5090 nu in:

```
5090 POKE 623,2
```

Na het runnen, beweegt de voetganger zich normaal voort, maar wordt de auto telkens slechts half weergegeven. Dat komt omdat de achterkant van de auto pm speler 2 is. Deze geeft in dit gemengde voorrangssysteem geen voorrang op de normale grafiek. Pm speler 1, de voorkant van de auto, heeft dat wel, en is daardoor wel zichtbaar.
Verander nu tenslotte regel 5090 in:

```
5090 POKE 623,8
```

U krijgt nu weer een gemengd voorrangsysteem. Alle pm spelers zitten in een groep. Deze heeft voorrang op een deel van de gewone grafiek, maar moet op zijn beurt, weer een deel van de gewone grafiek voorrang geven. De pm spelers worden achter het asfalt en de huizen getekend. Deze zijn getekend in de kleur van de kleurregisters 0 en 1. Maar de voetganger (een pm speler) wordt bovenop de lantaarnpalen getekend. De lantaarnpalen zijn getekend in de kleur van kleurregister 2.
Door de kleuren van de achtergrondgrafiek en de nummers van de spelers goed uit te kiezen is het mogelijk, door middel van het voor en achter elkaar langsschuiven van de verschillende figuren op het scherm, een zeer goede dieptesuggestie te bereiken.
Met behulp van het hiervoor gegeven programma kunt u gaan experimenteren.

Pm grafiek met verschillende kleuren

Hoewel elke pm speler slechts een enkele kleur kan hebben, is het zeer goed mogelijk een speler te maken die uit meer dan een kleur bestaat. Het nadeel daarvan is echter dat zo'n gekleurde speler samengesteld is uit meer dan een speler, elk in een andere kleur, en daardoor het aantal beschikbare spelers drastisch terugbrengt. Het resultaat is echter zeer mooi. Probeert u het volgende programma uit:

```
10 GRAPHICS 0
20 SETCOLOR 2,0,0
30 LIST
40 GOSUB 5000
50 X=90:Y=63
60 FOR I=SPELER0+Y TO SPELER0+Y+4
70 READ A:POKE I,A:NEXT I
90 FOR I=SPELER1+Y+6 TO SPELER1+Y+20
100 READ B:POKE I,B:NEXT I
120 FOR I=SPELER2+Y+17 TO SPELER2+Y+22
130 READ C:POKE I,C:NEXT I
140 POKE 704,15:REM KLEUR SPELER 0
150 POKE 705,118:REM KLEUR SPELER 1
160 POKE 706,54:REM KLEUR SPELER 2
170 POKE 53256,3
180 POKE 53257,3:POKE 53258,3
190 REM POSITIES
200 X=X-1:KL=INT(RND(0)*16)
210 IF X=0 THEN X=219
220 POKE 53248,X
230 POKE 53250,X
240 POKE 53249,X
250 POKE 704,KL
260 GOTO 200
270 END
499 REM DATA SPELER 0
500 DATA 29,53,114,105,64
509 REM DATA SPELER 1
510 DATA 71,71,69,68,68,124,127,255,255
515 DATA 255,153,153,153,129,129
519 REM DATA SPELER 2
520 DATA 102,102,102,102,102,102
5000 A=PEEK(106)-8
5010 POKE 54279,A
5020 PMBEGIN=256*A
5030 POKE 559,46
5040 POKE 53277,3
5050 SPELER0=PMBEGIN+512
```

```
5060 SPELER1=PMBEGIN+640
5070 SPELER2=PMBEGIN+768
5080 FOR I=PMBEGIN+512 TO PMBEGIN+896:POKE I,0
5085 NEXT I
5090 POKE 623,1
5100 RETURN
```

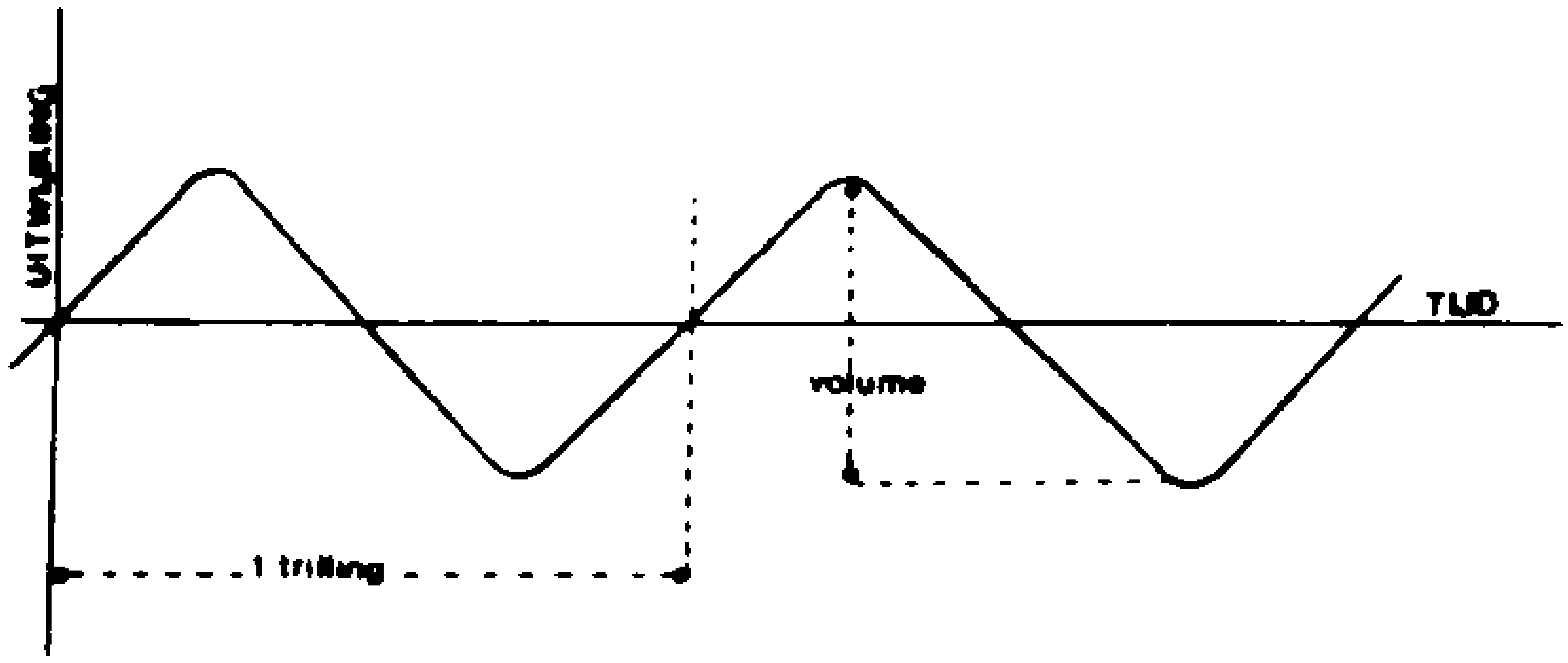
De truuk van dit programma is dat de drie pm spelers op elkaar geplaatst worden. Doordat elk van de spelers slechts een deel van de locomotief vormt, is het mogelijk de locomotief uit verschillende kleuren op te bouwen.
De regels 10-30 stellen het tekstscherm 0 in en zetten de listing van het programma op het scherm. Op deze wijze kunt u gelijk zien hoe een pm speler reageert op de listing.
Regel 50 bepaalt de beginpositie van de drie spelers.
De regels 60-130 lezen de gegevens voor de drie spelers in. Let op de factoren waarmee de Y-waarde in de regels 60, 90 en 120 wordt verhoogd. Alle vakjes van alle spelers zijn in de subroutine al op nul gezet en dus leeg. Alleen die vakjes die daadwerkelijk ingekleurd hoeven te worden, moeten van een gegeven uit de DATA lijst worden voorzien. Anders moeten er veel te veel gegevens onnodig worden ingetikt. De horizontale positie van elke speler is precies gelijk, maar de verticale positie verschilt.
De regels 140-160 bepalen de beginkleuren van de pm spelers. Alleen speler 0 (de rook) zal in de loop van het programma nog veranderen om de rook echter te doen lijken.
In regel 170 en 180 worden alledrie de spelers evenveel vergroot. Het is in principe mogelijk om een samengestelde speler te maken waarvan de onderdelen niet allemaal dezelfde vergrotingsfactor hebben (bijvoorbeeld een grof getekende auto in vier-dubbel formaat, met een fijn ingetekende chauffeur in normaal formaat) maar het vraagt enig rekenwerk.
In de regels 190-260 wordt de positie van de locomotief telkens uitgerekend en in de juiste geheugenlokaties gePOKEt. De kleur van speler 0 verandert telkens om de rook een beetje te laten 'verwaaien'.
De regels 499-520 bevatten de gegevens voor de verschillende spelers. De regels 5000-5100 bevatten de subroutine om de pm grafiek in te stellen.

20. GELUID EN MUZIEK

Zoals een schrijfmachine een bel heeft om de aandacht voor het einde van de regel te trekken, zo hebben computers al heel lang een eenvoudige geluidsmogelijkheid voor een 'beep'. Deze bliep is altijd bedoeld geweest om de aandacht te trekken van de gebruiker.
Met de opkomst van de spelletjes bleek dat zo'n eenvoudige piep niet voldoende was. Tegenwoordig beschikken de meeste microcomputers over een regelbare bliep, zodat eenvoudige geluiden (een-stems) gemaakt kunnen worden.
De Atari computers hebben verschillende geluidsgeneratoren (zeer fraaie piepers). *Met deze vier zeer veelzijdige geluidsgeneratoren is het mogelijk om een haast onbegrensde hoeveelheid verschillende geluiden te maken.* De mogelijkheden zijn zo groot dat de Atari niet alleen op verschillende manieren kan piepen, maar zeer fraaie muziek kan voortbrengen met vier stemmen tegelijk. In dit hoofdstuk zullen alleen de programmeerprincipes worden besproken, uiteraard vergezeld van de nodige voorbeelden. Omdat muziek zeer smaakgebonden is en het aantal mogelijkheden van de Atari te groot is om in een dik boek te zetten, zult u zelf door middel van experimenteren de leukste geluiden moeten zien te vinden.

Geluid

Geluid is het trillen van lucht. Dit trillen kan veroorzaakt worden door het trillen van een snaar (eventueel via een klankkast), het trillen van het vel van een trommel, het trillen van een rietje (in een fagot bijvoorbeeld) of op vele andere manieren. Het trillen van de lucht moet niet opgevat worden als op en neer trillen, maar als een opeenvolging van verdichtingen en verdunningen van de lucht. De lucht wordt achtereenvolgens even in elkaar geduwd en uit elkaar getrokken.
Op papier worden trillingen meestal weergegeven als een golf. Deze vorm is het resultaat van het tegen elkaar uitzetten van tijd en amplitude. De amplitude is het verschil tussen een verdichting en een verdunning. Er zijn twee basisgrootheden bij geluid: het volume en de frequentie van de trilling. De sterkte van het volume is hoorbaar. Op papier is het volume af te lezen aan de hoogte van de golf. Hoe hoger de golf hoe harder het geluid.
De frequentie is ook hoorbaar. Het zegt iets over de toonhoogte van het geluid. Hoe groter de frequentie, hoe hoger het geluid klinkt. De frequentie geeft aan hoe vaak een verdichting en verdunning in een seconde voorkomen. Men spreekt ook wel over het aantal trillingen per seconde. Dit wordt aangegeven in Hertz (Hz = aantal trillingen per seconde). Het menselijk oor kan geluiden van 30 Hz (een zeer lage brom) tot ongeveer 15000 Hz (een zeer hoge piep) horen. Het niveau van spreken en muziek ligt ongeveer tussen 100 en 4000 Hz.



De BASIC besturing

Het maken van geluid door de Atari kan door de gebruiker bepaald worden door de opdracht:

SOUND generator, hoogte, vorm, volume

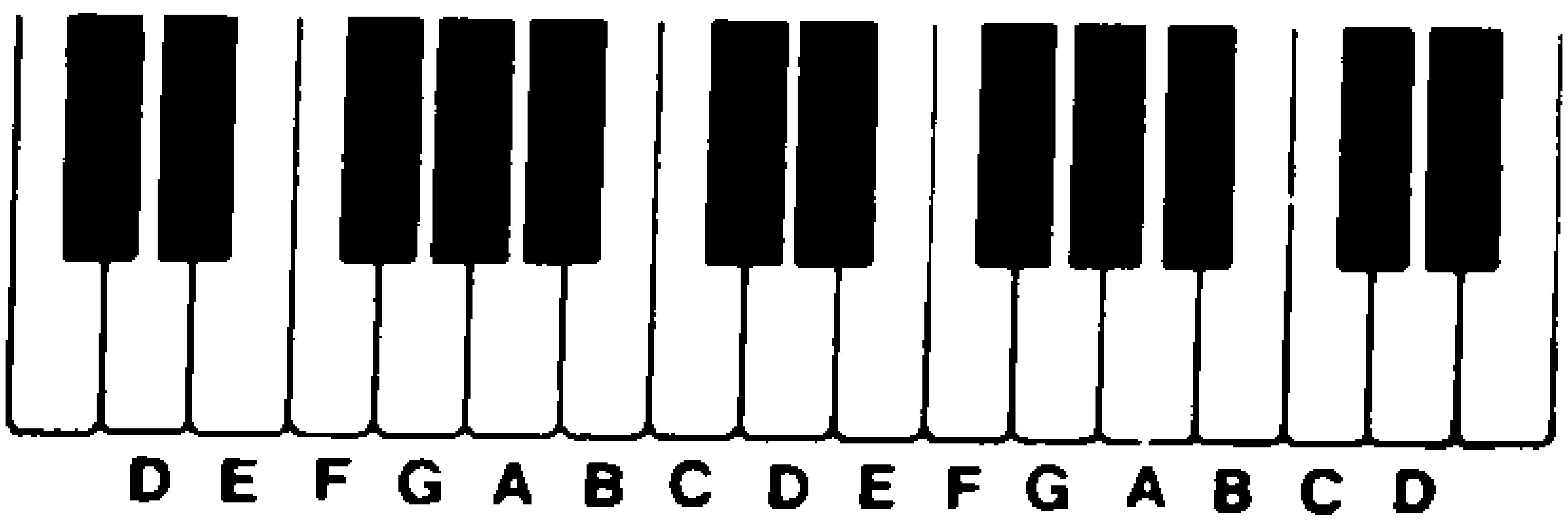
Deze opdracht werkt direkt op de geluidsgeneratoren. Het geluid kan individueel bepaald worden.
Achter SOUND moeten vier getallen, berekeningen of variabelen ingevuld worden.
De waarde 'generator' kan lopen van 0 tot en met 3. Deze waarde bepaalt welk van de vier geluidsgeneratoren het geluid gaat voortbrengen.
De waarde 'hoogte' bepaalt welke toonhoogte het voortgebrachte geluid zal hebben. Deze waarde moet liggen tussen 0 en 255. Hoe hoger de waarde, hoe hoger de toon. (zie hieronder)

De waarde 'vorm' bepaalt de vorm van de geluidsgolf:

0	- Bij lage toonhoogten krijgt u een witte ruis (dat is een hoog sissend geluid). Bij hoge toonhoogten krijgt u een pulserend geluid.
2 en 6	- Wisselend effect. Lijkt een beetje op een motor.
4	- Bij lage toonhoogten een pulserend geluid. Bij hoge toonhoogten een zoemtoon.
8	- Witte ruis. Wisselend van karakter bij wisselende toonhoogten. Geschikt voor explosies enzovoort.
12	- Zogenaamde naaldvormige pulsen. Ook geschikt voor muziek, maar de toonhoogten zoals opgegeven bij muziek kloppen nu niet meer.
10 en 14	- Bijna 'zuivere' tonen. Deze geluiden lijken het meeste op de normale tonen die door electronische geluidsinstrumenten gevormd worden.

De oneven waarden voor de vorm geven een klik als de SOUND opdracht gegeven wordt, en opnieuw een klik als de SOUND opdracht herhaald wordt om het geluid uit te schakelen.
De waarde 'volume' bepaalt hoe hard het geluid klinkt. Deze waarde moet tussen 0 en 15 liggen. Het uiteindelijke volume van het geluid wordt mede bepaald door de afstelling van de televisie of monitor. De waarde geeft meer een verhouding tussen met maximale en het minimale volume aan.
Als men meer dan één geluidsgenerator tegelijkertijd gebruikt is het raadzaam de waarden van de verschillende volumes samen niet groter te laten worden dan 32. Dit om vervormingen te voorkomen.

Eén van de taken van de SOUND opdracht is om uw Atari muziek te laten maken. U moet daarbij bedenken dat een deel van de weergave van de muziek afhankelijk is van uw televisie of monitor. Verschillende Atari's hebben geen eigen luidspreker, maar gebruiken die van uw televisie of monitor. Vindt u een bepaald geluid te hard of te zacht dan kunt u dat opvangen door de opdrachten te veranderen, maar ook door het volume van de televisie bij te stellen. Dit is mede afhankelijk van het feit of u het televisiegeluid normaal gesproken aan heeft staan voor de klikken van het toetsenbord en de weergave van de blieps.
Het toetsenbord van een piano kan als basis dienen bij het gebruiken van de SOUND opdracht. Bijna elk stuk dat u op een piano kunt spelen, kunt u ook door de Atari laten spelen. Daarnaast kan de Atari nog veel meer. Het is raadzaam om bij de bespreking van de verschillende voorbeelden telkens het gebruik van een piano voor ogen te houden. Nog mooier is het als u een piano bij de hand heeft. U kunt verschillende voorbeelden dan vergelijken met het geluid van een piano.

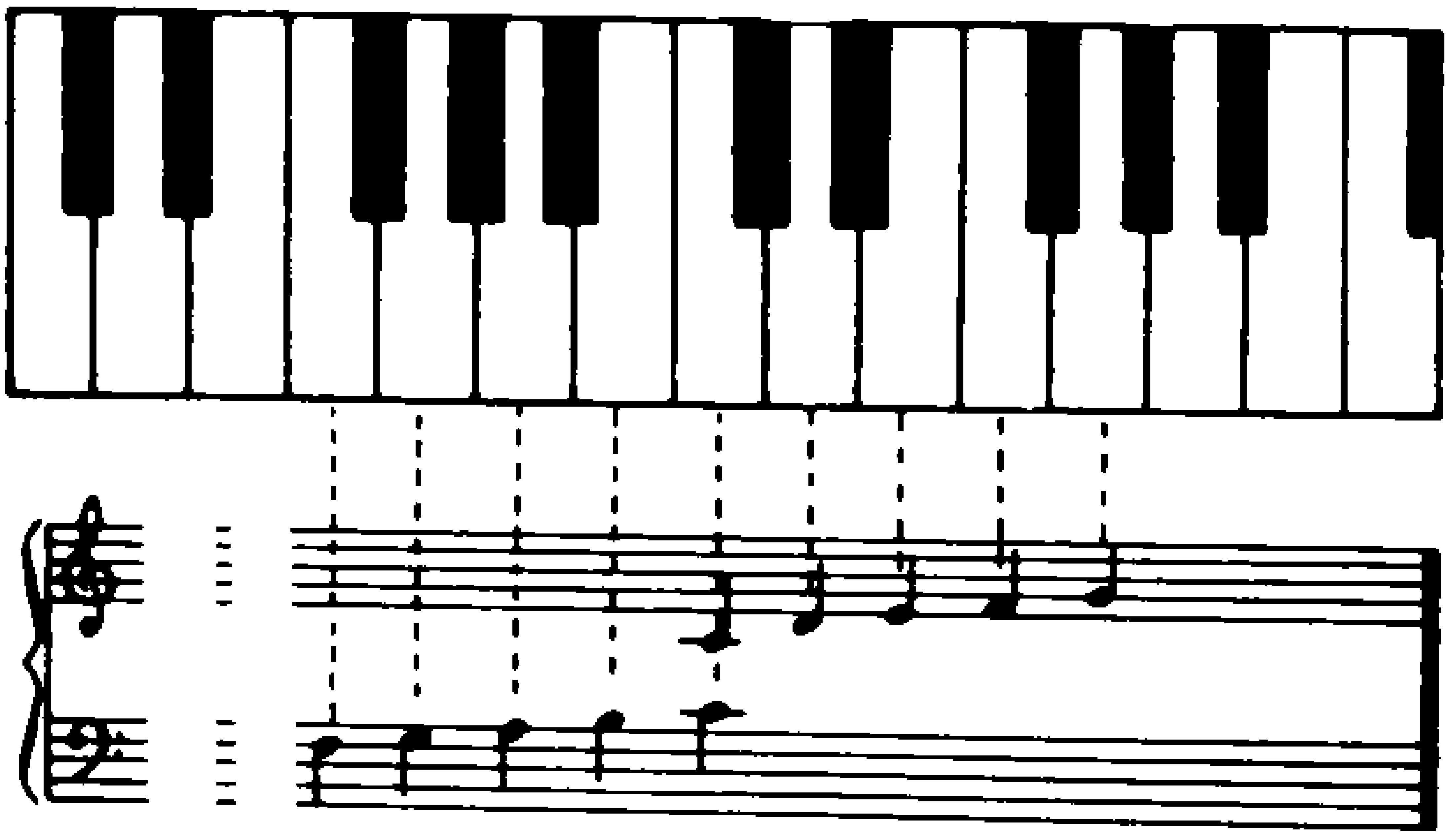


De toetsen van een piano geven verschillende noten aan, gegroepeerd in octaven. Een octaaf bestaat uit 8 hele noten die C, D, E, F, G, A ,B en C genoemd worden. De eerstvolgende noot na dit rijtje is de C van een octaaf hoger. Daarnaast kent de piano nog zwarte toetsen die halve noten aangeven (op papier aangegeven met #). Een octaaf kent dus: C, C#, D, D#, E, F, F#, G, G#, A, A#, B en C.

De SOUND opdracht heeft het toonbereik anders aangegeven. Om een bepaalde noot te spelen moet u de juiste toonhoogte opgeven.

toonhoogte	noot	toonhoogte	noot
255	B	81	G
243	C	76	G#
230	C#	72	A
217	D	68	A#
204	D#	64	B
193	E	60	C
182	F	57	C#
173	F#	53	D
162	G	50	D#
153	G#	47	E
144	A	45	F
136	A#	42	F#
128	B	40	G
121	C	37	G#
114	C#	35	A
108	D	33	A#
102	D#	31	B
96	E	29	C
91	F	28	C#
85	F#	26	D

De Atari kent drieeneenhalve octaaf. In de tabel hierboven zijn de halve noten aangegeven door een # teken.



Muziek

Nu de verschillende noten vastliggen is het mogelijk om muziek te maken. Een eenvoudige manier is het maken van een lijstje met waarden die de Atari moet afwerken. Het volgende programma laat alle aanwezige noten horen:

```
10 GRAPHICS 0
20 FOR A=1 TO 40
30 READ HOOGTE
40 SOUND 1,HOOGTE,14,12
50 FOR WA=1 TO 90:NEXT WA
60 SOUND 1,0,0,0
70 FOR WA=1 TO 30:NEXT WA
80 NEXT A
100 DATA 255,243,230,217,204,193,182,173
105 DATA 162,153,144,136,128,121,114,108,102
110 DATA 96,91,85,81,76,72,68,64,60,57,53
115 DATA 50,47,45,42,40,37,35,33,31,29,28,26
```

Op dezelfde wijze is het mogelijk een wijsje te maken:

```
10 GRAPHICS 8
20 POKE 14,0:POKE 15,152
30 FOR N=0 TO 1023:POKE 38912+N,0
40 NEXT N
50 FOR Q=512 TO 765
60 POKE 38912+Q,Q-511
70 NEXT Q
80 POKE 559,46
90 POKE 704,1
100 POKE 53248,40
110 POKE 53256,0
120 POKE 53277,3
130 GOTO 130
```

Het is weliswaar leuk om een computer een wijsje te horen spelen, maar het klinkt niet beter dan een beginnende pianoleerling bij les 1. Heel snel al leert elke muziekmaker dat niet elke noot even lang moet klinken. Noten worden onderverdeeld in:

L1	hele noten
L2	halve noten
L4	kwart noten
L8	achtste noten
L16	zestiende noten
L32	tweeendertigste noten

De lengte van de noten is niet zo maar te bepalen bij de SOUND opdracht. De SOUND opdracht stelt alleen een geluidsgenerator in. Zodra deze ingesteld is, klinkt het gevraagde geluid. Om het geluid te stoppen is een END opdracht nodig, of een SOUND opdracht die alle waarden die bij een bepaalde generator horen, weer op 0 instelt.
De lengte van de tonen is echter zeer redelijk te bepalen door middel van de FOR..NEXT lus. Door een FOR..NEXT lus tot 32 te laten lopen bij elke hele noot, zal het éénmaal laten doorlopen van de lus zorgen voor één tweeendertigste noot. Het volgende programma geeft hiervan een voorbeeld.

```
10 GRAPHICS 0
20 FOR Q=1 TO 2
30 RESTORE
40 FOR A=1 TO 44
50 READ HOOGTE,DUUR
60 SOUND 1,HOOGTE,14,12
70 FOR WA=1 TO 70*DUUR:NEXT WA
80 SOUND 1,0,0,0
90 FOR WA=1 TO 3:NEXT WA
100 NEXT A
110 NEXT Q
120 DATA 60,2,53,1,47,2,53,2,40,2,81
125 DATA 2,81,4,72,2,64,1,60,2,64,2
130 DATA 60,2,47,2,53,4,64,2,60,1,53,2,64,2,53
135 DATA 2,47,2,45,2,53,2,81,2,81,2,60,2,53,2
140 DATA 47,2,40,2,40,2,45,2,47,2,53,2,60
145 DATA 2,81,2,47,2,53,1,60,4,60,2
150 DATA 60,2,35,2,35,2,40,2,64,2,60,4
```

In regel 50 worden voor elke te spelen noot zowel de toonhoogte als de duur gelezen uit de datalijst. De toonhoogte wordt verwerkt in de SOUND opdracht in regel 60. De duur van de toon wordt verwerkt in een FOR..NEXT lus in regel 70. Dit is eigenlijk niets anders als een vertragingslus.
Wilt u het wijsje als een doorlopende opeenvolging van klanken horen dan kunt u regel 80 en 90 verwijderen en regel 115 toevoegen:

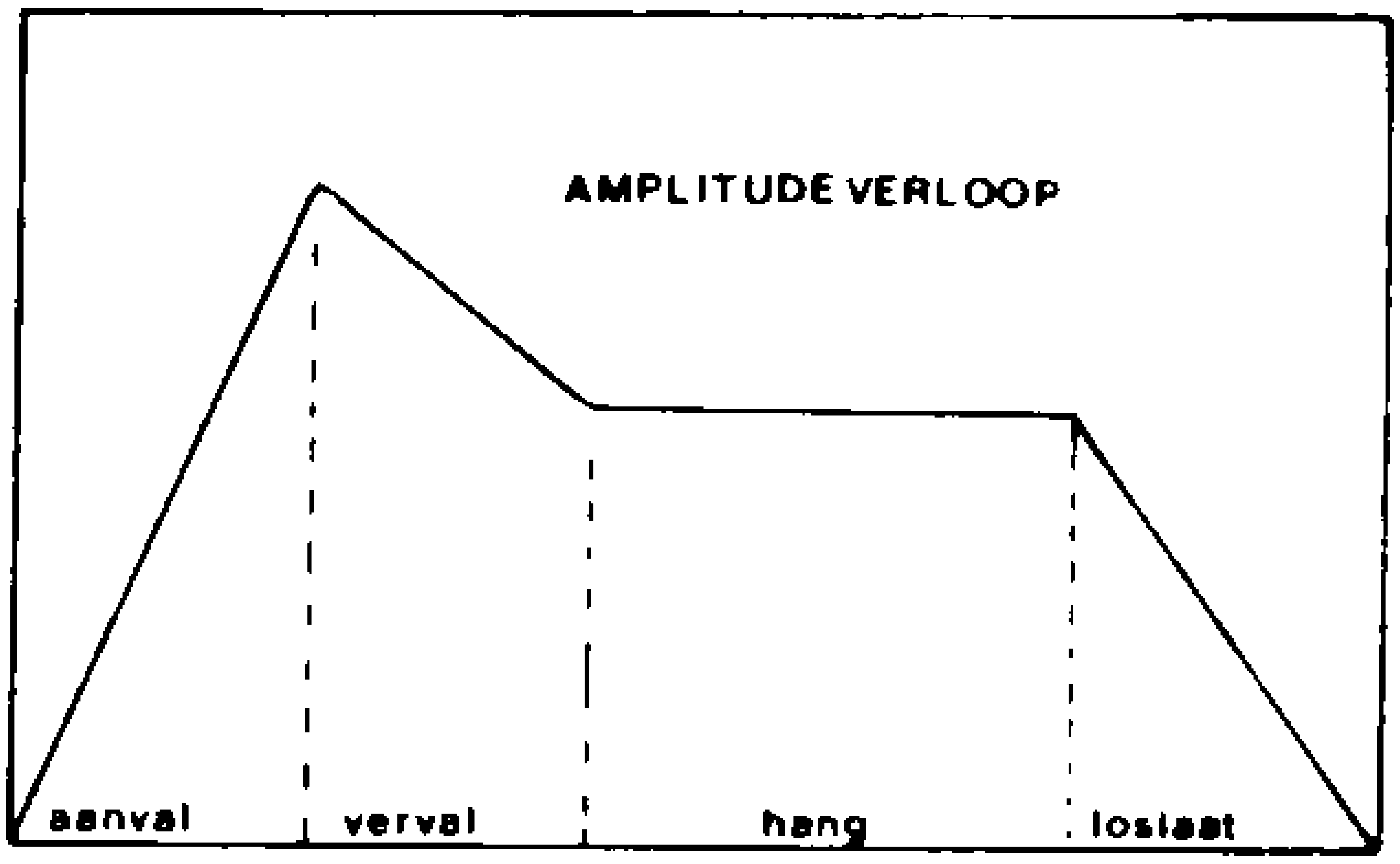
```
115 FOR WA=1 TO 420 : NEXT WA
```

U hoort nu onmiddellijk waarom de korte rustpauze tussen elke twee noten nodig is.

Omhullenden

Een omhullende is de manier waarop het volume verandert tijdens het klinken van een toon. Bij de Atari is dit niet zomaar te regelen. Bij het instellen van de SOUND opdracht is slechts een volume-instelling mogelijk. Door de SOUND

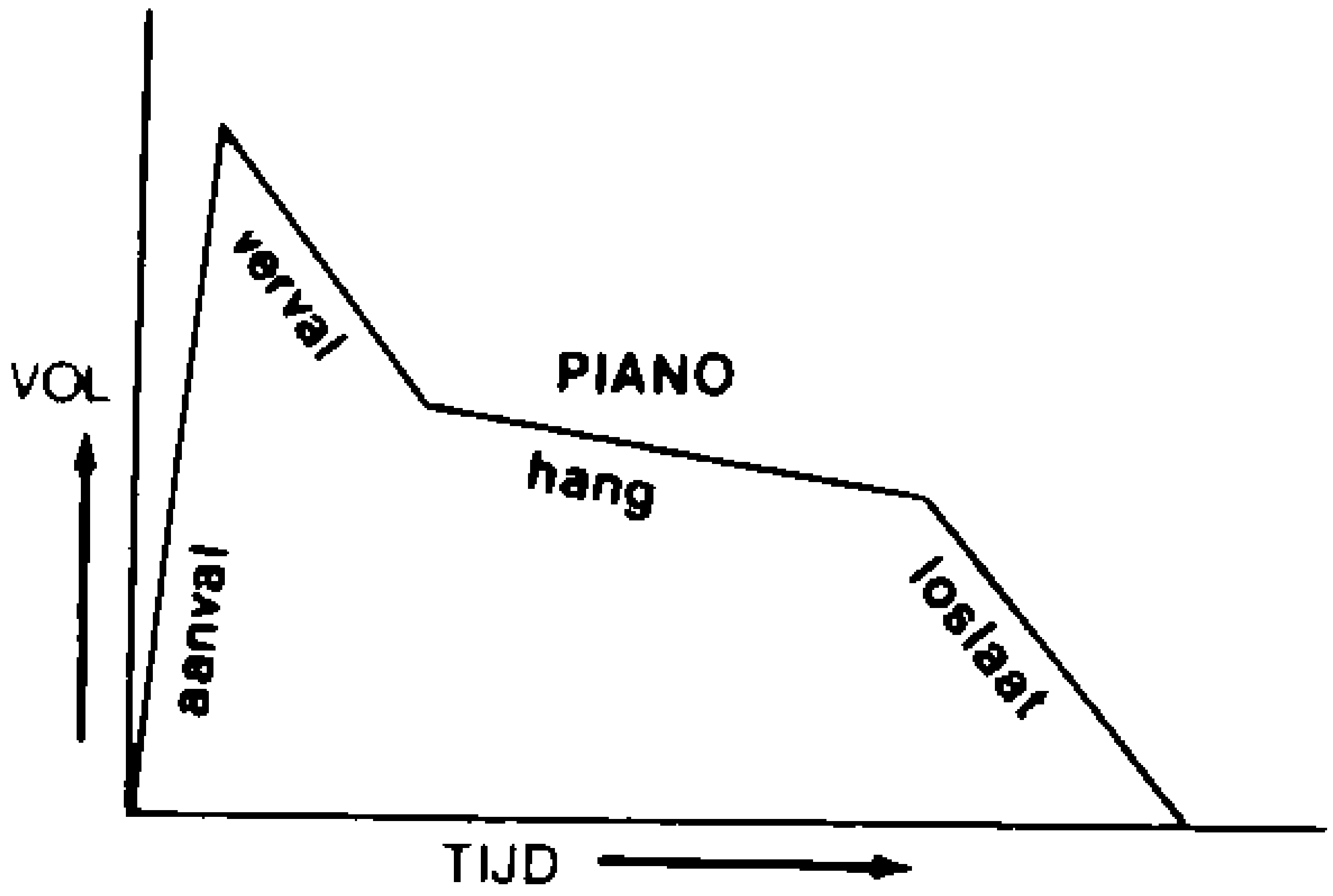
opdracht te beeindigen (door het opgeven van een 0 voor het volume, of door een END of RUN opdracht), wordt het volume weer uitgeschakeld.
Een omhullende (ook wel, naar het engels, een envelope genoemd) kent in het algemeen vier fasen:



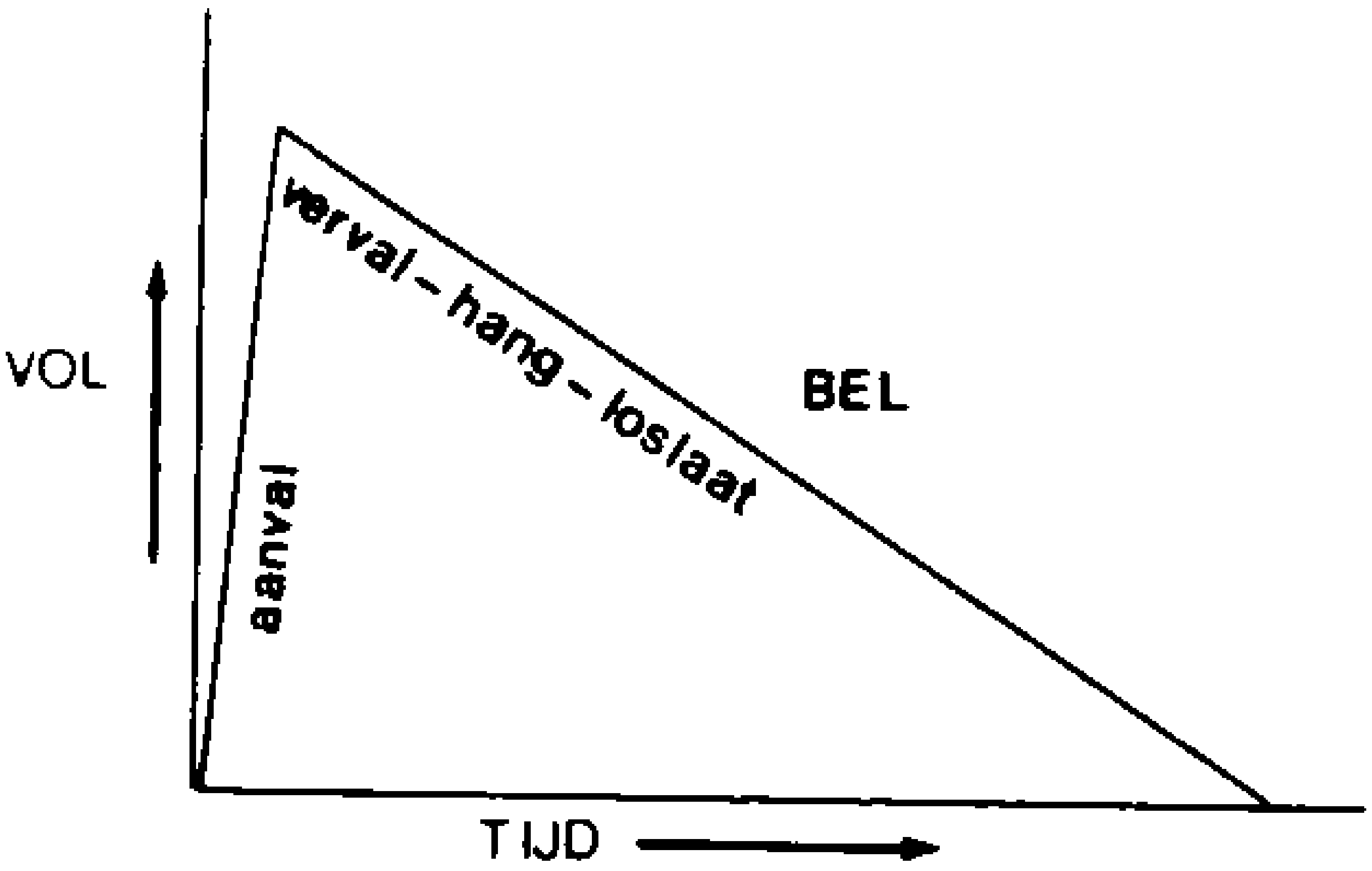
Te onderscheiden zijn:

- de aanval-fase. Dit is de fase waarin het geluid snel tot zijn grootste volume komt.
- de verval-fase. Dit is de fase waarin het geluid terugvalt tot zijn normale volumeniveau.
- De hang-fase. Dit is de fase waarin het geluid op normale sterkte klinkt.
- De loslaat- of uitsterffase. Dit is de fase waarin het geluid weg sterft.

De fasen zijn bij elk instrument anders. Het is het verschil van de omhullende dat een van de karaktereigenschappen van instrumenten bepaalt. Zo heeft een piano een snel verlopende aanvalfase. Op het moment dat de vinger een pianotoets indrukt, raakt een hamertje de snaar en stijgt het volume van de voort te brengen toon heel snel van nul naar maximaal volume. Direkt daarna zwakt het geluid een heel klein beetje af tot het volumeniveau dat een tijdje zal blijven. De vervalfase is gematigd snel. Na het verval blijft de snaar nog een tijd trillen en geluid voortbrengen. Zolang de pianospeler de toets ingedrukt houdt, wordt de snaar niet gedempt, en sterft het geluid tijdens deze hangfase slechts zeer langzaam weg. Op een gegeven moment laat de painospeler de toets los, de snaar wordt gedempt en het geluid sterft snel weg: de loslaatfase is wat sneller. De omhullende van een piano ziet er als volgt uit:



De omhullende van instrumenten als een bel, triangel en xylophoon zijn weer heel anders. Er is een zeer korte aanvalsfase wanneer de hamer of de stok het instrument raakt. Daarna volgt een zeer lang en zeer gelijkmatig afzwakken van het geluid. De verval-, hang- en loslaatfase vallen eigenlijk samen.



Zoals gezegd heeft een omhullende niet met de SOUND opdracht te maken. Door een paar FOR..NEXT lussen te gebruiken in samenhang met de waarde voor het volume in de SOUND opdracht, is toch een omhullende te vormen. Het volgende programma laat horen hoe een Atari-piano klinkt.

```
10 GRAPHICS 0
20 DUUR=4
30 FOR Z=1 TO 44
40 READ HOOGTE
50 GOSUB 500
60 NEXT Z
70 END
100 DATA 204,193,182,173,162,153,144,136,128,121
105 DATA 114,108,102,96,91,85,81,76,72,68,64
110 DATA 60,57,53,50,47,45,42,40,37,35,33,31,29
```

```
115 DATA 28,26,28,29,31,33,35,35,33,33
499 REM PIANO
500 FOR AA=1 TO 10 STEP 5:SOUND 0,HOOGTE,14,AA:NEXT AA
510 FOR VE=15 TO 12 STEP -1:SOUND 0,HOOGTE,14,VE
515 FOR WA=1 TO 10:NEXT WA:NEXT VE
520 FOR WA=1 TO 60*DUUR:NEXT WA
530 FOR KK=12 TO 0 STEP -0.5
535 SOUND 0,HOOGTE,14,KK:NEXT KK
540 SOUND 0,0,0,0
550 RETURN
```

De regels 20-60 zorgen voor het lezen van een toonwaarde, en het door een subroutine laten klinken van de toon.
De regels 110 en 110 bevatten de lijst met toonwaarden.
De subroutine vanaf regel 500 zorgt voor een piano-omhullende:
In regel 500 gaat het volume zeer snel omhoog. Dit is de aanvalsfase.
In regel 510 gaat het volume weer een beetje naar beneden: de vervalfase.
In regel 520 blijft het volume een tijdje constant: de hangfase. Hoe lang de hangfase duurt is afhankelijk van de waarde van de variabele "DUUR". Door hier een variabele te gebruiken, is het -ook met een omhullende- mogelijk de lengte van de individuele tonen te regelen, en daardoor een wijsje te maken.
In regel 530 sterft het geluid uit: de loslaatfase.
Regel 540 is eigenlijk niet meer nodig. Deze is erbij gezet voor die gevallen waarin de loslaatfase niet helemaal tot 0 gaat, maar de toon wel uitgezet moet worden.
Als u wilt dat de pianist wat vloeiender speelt, moet u regel 540 laten vervallen en regel 530 aanpassen:

```
530 FOR KK=12 TO 4 STEP -0.5:SOUND 0,HOOGTE,14,KK:NEXT KK
```
of
```
530 FOR KK=12 TO 7 STEP -0.5:SOUND 0,HOOGTE,14,KK:NEXT KK
```

Op dezelfde wijze kunt u een xylophoon laten horen:

```
10 GRAPHICS 0
20 DUUR=4
30 FOR Z=1 TO 30
40 READ HOOGTE
50 GOSUB 500
60 NEXT Z
70 END
100 DATA 91,85,81,76,72,68,64
110 DATA 60,57,53,50,47,45,42,40,37,35,33
120 DATA 31,29,28,26,28,29,31,33,35,35,33,33
499 REM BEL
500 FOR AA=1 TO 10 STEP 5
```

```
505 SOUND 0,HOOGTE,14,AA:NEXT AA
510 FOR VE=15 TO 6 STEP -0.05
520 SOUND 0,HOOGTE,14,VE:NEXT VE
530 RETURN
```

Meer dan een kanaal

Het is mogelijk de verschillende kanalen die de SOUND opdracht kent, tegelijk te gebruiken. De kanalen zijn genummerd van 0 tot en met 3. Elk kanaal moet afzonderlijk door een SOUND opdracht gestuurd worden. Indien gewenst, moet elk kanaal ook afzonderlijk weer uit gezet worden. Een END opdracht sluit alle vier de kanalen af.
Het volgende programma laat u het slaan van een klok horen. Merk op hoevol echter het geluid klinkt, nu er vier kanalen tegelijk gebruikt worden.

```
5 FOR Q=1 TO 12
10 FOR VOLUME=15 TO 2 STEP -0.5
20 SOUND 0,74,10,VOLUME
30 SOUND 1,34,10,VOLUME
40 SOUND 2,57,10,VOLUME
50 SOUND 3,48,10,VOLUME
60 FOR WA=1 TO 10:NEXT WA
70 NEXT VOLUME
80 NEXT Q
85 FOR WA=1 TO 300:NEXT WA
90 END
```

U krijgt weer een heel ander effect als u de vier kanalen niet tegelijk laat beginnen, maar laat verspringen. Voeg de volgende regels aan bovenstaand programma toe:

```
25 FOR WA=1 TO 25 : NEXT WA
35 FOR WA=1 TO 25 : NEXT WA
45 FOR WA=1 TO 25 : NEXT WA
55 FOR WA=1 TO 25 : NEXT WA
```

Probeert u zelf om een stuk, dat geschreven is voor bijvoorbeeld een piano en waarbij meer dan één toon tegelijk gespeeld moet worden, om te zetten voor de Atari. U zult daarbij van elke toon drie gegevens op moeten geven: toonhoogte, tijdsduur (in maten!) en kanaal. Probeert u daarbij elke toon met een omhullende te laten spelen. Het hoeft geen echt lang programma te zijn om al een wonderbaarlijk fraai effect te geven.
PS: in veel muziek worden hele stukken regelmatig herhaald. Maak gebruik van de RESTORE opdracht, in samenhang met het telkens op een nieuwe DATA regel laten beginnen van een nieuw stuk, om overbodige DATA invoer te beperken.

Naast de welluidende muziek die u aan de hand van de hiervoor gegeven voorbeelden al gemaakt heeft, komt het vaak van pas over enkele geluidseffecten te beschikken. Zeker in computerspelletjes is geluid een belangrijke factor. U vindt hieronder een stel voorbeelden van geluidseffecten. Er is geen algemene leidraad te geven die u kunt volgen bij het maken van nieuwe geluidseffecten. U zult door uitproberen nieuwe effecten moeten vinden. De voorbeelden helpen u op weg.

Een bal

Dit geluidseffect moet u nog bekend voorkomen van een van de simulatieprogramma's eerder in dit boek. U kunt het nu als los geluidseffect gebruiken.

```
10 FOR B=20 TO 1 STEP -1
20 FOR A=1 TO 4
30 SOUND 0,125,14,6
40 NEXT A
50 SOUND 0,0,0,0
60 FOR A=1 TO B*6:NEXT A
70 NEXT B
```

Lucky Luke

Snel achter elkaar vuurt onze held zes schoten af. De boef is al zeker van zijn zaak. Maar.... Lucky Luke is de enige man in het wilde westen met zeven kogels in zijn pistool.

```
10 FOR A=1 TO 6
20 SOUND 0,5,0,15
30 FOR WA=1 TO 25:NEXT WA
40 SOUND 0,0,0,0
50 FOR WA=1 TO RND(0)*600+100:NEXT WA
60 NEXT A
70 FOR WA=1 TO 900:NEXT WA
80 SOUND 0,5,0,15
90 FOR WA=1 TO 25:NEXT WA
100 END
```

Safari geluiden

Overal waar u door het manshoge gras loopt, wordt u begeleid door het snerpen van de krekels.

```
10 FOR A=1 TO 100
20 FOR B=3 TO 6
30 SOUND 0,B,10,8
40 SOUND 1,B-2,10,10
50 SOUND 2,B+5,10,6
```

```
60 NEXT B
70 NEXT A
80 END
```

In het tropisch regenwoud klinkt het weer anders. Daar hoort u 's ochtends vroeg het gekwetter van de vogels.

```
10 FOR A=1 TO 800
20 FOR B=3 TO RND(0)*19
30 SOUND 0,B,10,11
40 NEXT B
50 NEXT A
60 END
```

Maar pas op voor de leeuwen, want voordat u het weet:

```
10 FOR A=1 TO 600
20 FOR B=-110 TO 110 STEP 3
30 SOUND 0,ABS(B)+25,10,8
40 NEXT B
50 NEXT A
```

en moet u snel waar huis, waar -veel vertrouwder- klinkt:

```
10 FOR A=1 TO 100
20 SOUND 0,53,10,8
30 FOR WA=1 TO 200:NEXT WA
40 SOUND 0,74,10,8
50 FOR WA=1 TO 200:NEXT WA
60 NEXT A
70 END
```

Take-off

Tot slot een vertrouwd computergeluid; het stijgen van een raket.

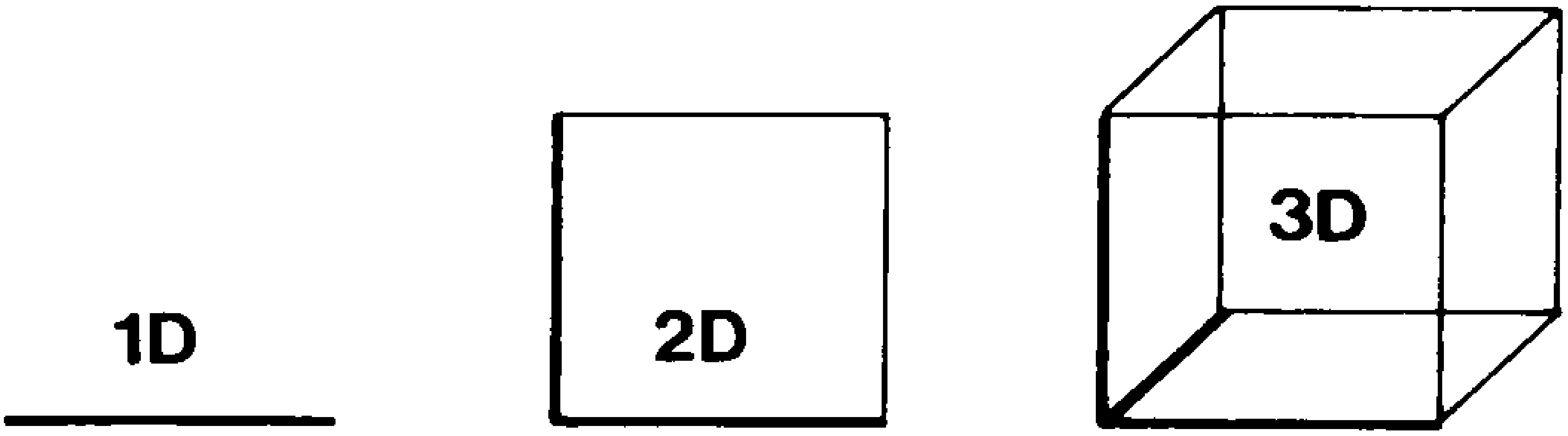
```
10 FOR B=0 TO 45
20 SOUND 0,B,8,B/3
30 NEXT B
40 FOR B=45 TO 3 STEP -1
50 SOUND 0,B,8,B/5
60 FOR WA=1 TO 50+B*3:NEXT WA
70 NEXT B
80 FOR WA=1 TO 900:NEXT WA
90 END
```

21. DE DERDE DIMENSIE

Een, twee of drie

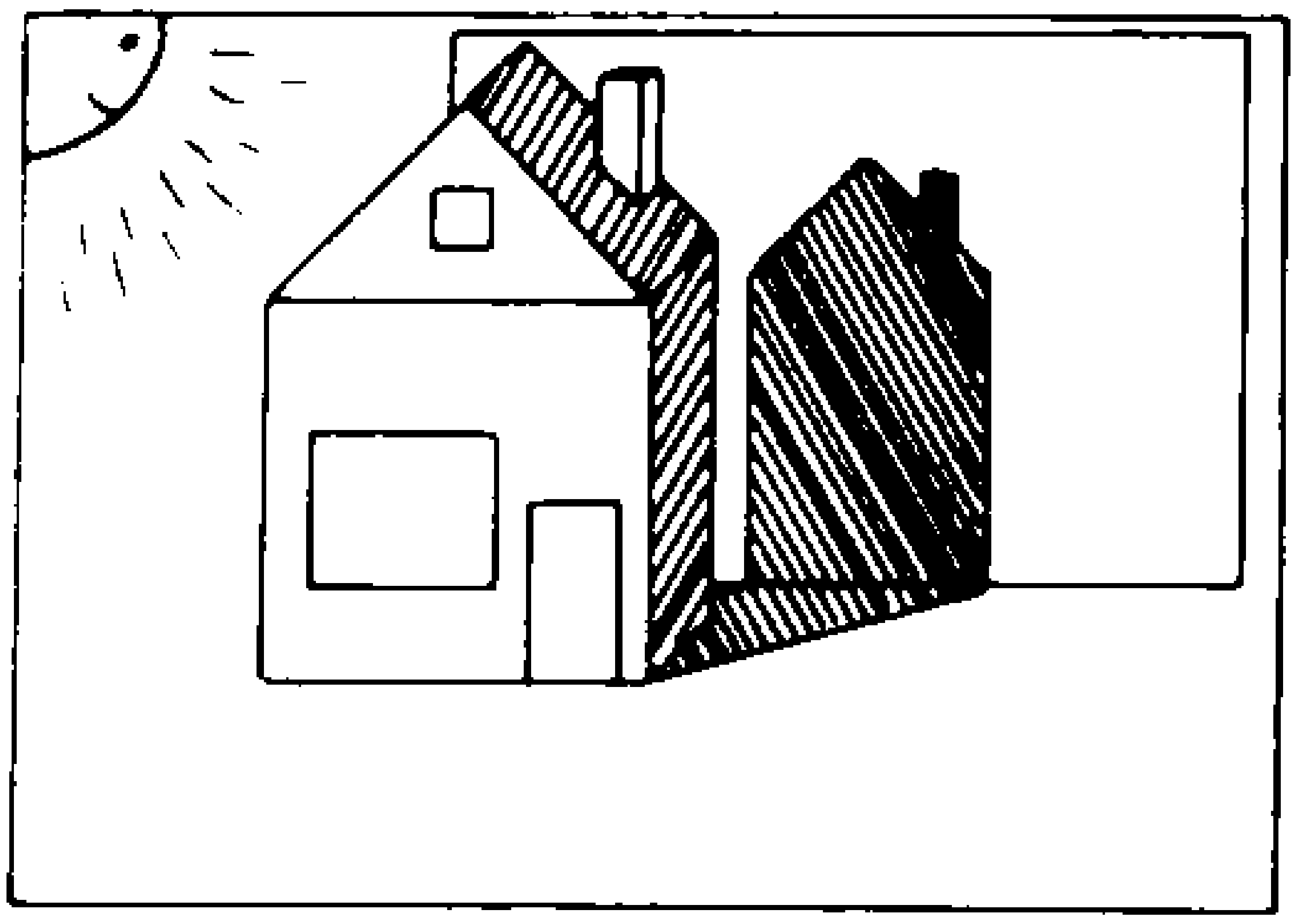
Het beeldscherm van uw televisie of monitor is plat. Het papier waarop de tekst die u nu leest gedrukt is, is plat. Men zegt van deze voorwerpen dat zij in twee dimensies uitgedrukt kunnen worden.
De normale voorstelling van de verschillende dimensies is:

Eerste dimensie: een richting. Een lijn.
Tweede dimensie: twee richtingen. Een vlak.
Derde dimensie: drie richtingen. Een ruimte.



Een dimensie, twee dimensies, drie dimensies

Een één-dimensionaal object of een twee-dimensionaal object kan gemakkelijk op papier getekend worden, zonder dat er ook maar iets aan de vorm veranderd hoeft te worden. Zo is het ook altijd duidelijk met een (twee-dimensionale) tekening te zien, welk één- of twee-dimensionaal object wordt bedoeld.
Om niet al te langdradig te worden korten we een-dimensionaal af door **1D**, twee-dimensionaal door **2D** en drie-dimensionaal door **3D.**
Zodra een object weergegeven moet worden op een voorwerp met een lagere dimensie krijgen we te maken met een probleem. Er moet dan geprojecteerd worden. Dit projecteren moet u heel letterlijk opvatten. Het werkt net zo als bij fotografie. Een 3D object, bijvoorbeeld een huis, wordt geprojecteerd op een 2D object, namelijk de film in het fototoestel. Een ander dagelijks voorkomend voorbeeld is de schaduw van een 3D object, bijvoorbeeld een huis, op een 2D object, bijvoorbeeld de grond of een muur.



drie-dimensionaal naar twee-dimensionaal

Een van de problemen van projecteren is dat na de projectie de diepte van de oorspronkelijke vorm niet af te lezen is in de projectie. Met deze problemen krijgt u te maken als u 3D tekeningen met de Atari wilt maken. *Het beeldscherm waar de tekening uiteindelijk op komt te staan is immers plat. Er zal geprojecteerd moeten worden.*
De programma's in dit hoofdstuk zullen bijna allemaal tekeningen maken die alleen de buitenste lijnen van een voorwerp (de ribben) aangeven. Er is echter een methode die min of meer **exclusief** voor de Atari is, en waarmee diepte gesuggereerd kan worden.

Dieptesuggestie door schaduwwerking

Een mens herkent een tekening als een voorstelling van een ruimtelijk object door verschillende aanwijzingen. Een van die aanwijzingen is de manier waarop de ribben van het getekende voorwerp lopen. Een andere aanwijzing is de manier waarop het licht op het voorwerp valt. Zo zal een kubus die onder een lamp staat, een lichte bovenkant hebben, een gematigd lichte voorkant en een donkere zijkant. Door nu drie vlakken te tekenen en deze vlakken een verschillende tint te geven, kan de diepte worden gesuggereerd . Deze methode zal bij de meeste computers niet lukken, omdat die slechts een beperkt aantal kleuren kunnen weergeven. Het grote voordeel van de Atari is dat er behalve een aantal kleuren ook van elke kleur een groot aantal helderheidsgraden kunnen worden getoond. Het volgende programma maakt hier gebruik van.

```
10 GRAPHICS 7+16
20 OPEN #1,4,0,"K:"
30 FOR Q=0 TO 4:SETCOLOR Q,9,4:NEXT Q
40 X=40:Y=40
50 COLOR 1
60 FOR Z=0 TO 45
```

```
70 PLOT X,Y+Z:DRAWTO X+45,Y+Z
80 NEXT Z
90 COLOR 2
100 FOR Z=1 TO 19
110 PLOT X+Z,Y-Z:DRAWTO X+Z+45,Y-Z
120 NEXT Z
130 COLOR 3
140 FOR Z=1 TO 19
150 PLOT X+45+Z,Y-Z:DRAWTO X+45+Z,Y+45-Z
160 NEXT Z
170 FOR Z=0 TO 2
180 GET #1,KL
190 SETCOLOR Z,1,3*KL
200 NEXT Z
210 GOTO 170
```

In regel 10 wordt grafisch scherm 7 zonder tekstvenster geopend. Regel 20 opent een invoerkanaal voor het toetsenbord. Op deze wijze kan een GET opdracht, later in het programma, gegevens van de gebruiker krijgen.
In regel 30 worden alle kleurregisters ingesteld op de **achtergrondkleur.** De kubus kan nu getekend worden, zonder dat de gebruiker ook maar iets ziet. Om de kubus later in de loop van het programma van kleur te veranderen is het niet nodig de kubus helemaal opnieuw te tekenen. Het is voldoende de kleuren in de kleurregisters te veranderen.
De regels 50-80 tekenen de voorkant van de kubus.
De regels 90-120 tekenen de bovenkant van de kubus.
De regels 130-160 tekenen de zijkant van de kubus.
Door de regels 170 en 210 worden telkens drie waarden van het toetsenbord gelezen.
Regel 180 neemt telkens de waarde van een ingedrukte toets.
Regel 190 zorgt ervoor dat deze waarde wordt omgezet in een kleur van een van de vlakken.
Regel 210 zorgt ervoor dat u dit proces eindeloos kunt herhalen.
Run het programma en tik achter elkaar in: 5, 4, en 3 of G, F en H of de wortelfunktie S, Q en R. U kunt ook elke andere combinatie van toetsen indrukken. Probeer zelf uit welke combinaties fraai zijn.

Hole in one

U kunt ook gebogen vlakken een diepte-suggestie geven. Daarvoor komt schermmodus 9, die 16 helderheden toestaat, goed van pas. Probeer het volgende programma om te zien waar die Amerikanen met hun club altijd maar weer tegenaan slaan.

```
10 GRAPHICS 9
12 DIM HOEK(18,2)
14 DEG :FOR A=0 TO 360 STEP 20
16 HOEK(A/20,1)=COS(A):HOEK(A/20,2)=SIN(A)
18 NEXT A
20 X=40:Y=80
30 STRAAL=35
50 FOR KL=1 TO 15 STEP 0.12
60 COLOR KL
70 PLOT X+STRAAL*0.5,Y
80 FOR Z=0 TO 18
90 DRAWTO STRAAL*HOEK(Z,1)*0.5+X,2*HOEK(Z,2)*STRAAL+Y
100 NEXT Z
110 STRAAL=STRAAL-0.3:IF STRAAL<0 THEN STRAAL=0
120 X=X+0.07
130 NEXT KL
140 GOTO 140
```

In de regels 12-18 worden de waarden die normaal gesproken bij het vormen van een cirkel worden uitgerekend, allemaal in een array gestopt. Dit programma maakt een heleboel cirkels. Door de verschillende sinussen en cosinussen van te voren vast uit te rekenen en de uitkomsten in een array te plaatsen, kan de computer voor alle cirkels volstaan met het opzoeken van de juiste waarde in de array. Dit is vele malen sneller dan het telkens opnieuw uitrekenen van de gewenste waarde.
De opdracht

DEG

in regel 14 zorgt ervoor dat de Atari in graden gaat rekenen (360 graden in een cirkel). Zonder deze opdracht rekent de Atari met radialen. Daarvan gaan er 2*PI (ongeveer 2*3.14) in een cirkel.
Regel 20 stelt het begin-middelpunt in.
De regels 50-130 tekenen een heel stel cirkels. Elke cirkel is een klein beetje kleiner dan de vorige, en de kleur is iets lichter. Het middelpunt verschuift bij elke cirkel naar links.
Het uiteindelijke effect is een golfbal met een sterke dieptesuggestie door imitatie van opvallend licht.

Tunnels

De dieptesuggestie die u kunt oproepen door het maken van tunnels heeft veel te maken met gezichtsbedrog. Door een aantal gelijkvormige figuren steeds een beetje kleiner binnen elkaar te tekenen, ontstaat de suggestie van een tunnel. Dit soort tekeningen zijn eenvoudig met papier en potlood te maken, maar dat kost erg veel tijd. Een computer is bij uitstek geschikt voor dit soort

tekeningen, omdat zij bestaan uit het vele malen herhalen van een gelijksoortige handeling. Probeer eerst het volgende programma.

```
10 GRAPHICS 8+16
20 XR=0:XL=319
30 YB=0:YO=191
40 DXR=+5:DXL=-3
50 DYB=3:DYO=-2
60 PLOT XR,YB
70 DRAWTO XR,YO
80 DRAWTO XL,YO
90 DRAWTO XL,YB
100 DRAWTO XR,YB
110 XR=XR+DXR:XL=XL+DXL
120 YB=YB+DYB:YO=YO+DYO
130 IF XL<=XR+50 OR YO<=YB THEN 130
140 GOTO 60
```

In de regels 20 en 30 worden vier waarden opgegeven. Met deze vier waarden is een rechthoek te tekenen.
Door de waarden steeds een beetje te verkleinen, en de erbij horende rechthoek te tekenen, ontstaat een stelsel van rechthoeken. Dit stelsel geeft de suggestie van een tunnel.
Doordat de verkleiningen voor de vier waarden verschillen, loopt de tunnel een beetje scheef weg. Daardoor wordt de dieptesuggestie nog groter.
Door gebruik te maken van een SIN funktie kan gesuggereerd worden dat de tunnel 'golft'.

```
10 GRAPHICS 8+16
20 XR=10:XL=310
30 YB=10:YO=181
40 DXR=5:DXL=-3
50 DYB=ABS(SIN(YB)*3):DYO=ABS(SIN(YO))*-3
60 PLOT XR,YB
70 DRAWTO XR,YO
80 DRAWTO XL,YO
90 DRAWTO XL,YB
100 DRAWTO XR,YB
110 XR=XR+DXR:XL=XL+DXL
120 YB=YB+DYB:YO=YO+DYO
130 IF XL<=XR+50 OR YO<=YB THEN 130
140 GOTO 50
```

Op dezelfde wijze kan er ook voor opstapjes in de tunnel gezorgd worden. Verander:

```
50 DYB=ABS(SIN(YB/9)*3):DYO=ABS(SIN(YO/9))*-10
130 IF XL<=XR OR YO<=YB THEN 130
```

Voor de tweeter-luidspreker van een grote geluidsinstallatie verandert u:

```
50 DYB=ABS(SIN(YB/50))*3:DYO=ABS(SIN(YO/50))*-10
```

U hoeft zich natuurlijk niet te beperken tot rechthoekige tunnels:

```
10 GRAPHICS 8+16
20 XR=10:XL=310
30 YB=10:YO=181
40 DXR=8:DXL=-6
50 DYB=4:DYO=-2
60 PLOT (XR+XL)/2,YB
70 DRAWTO XR,YO
80 DRAWTO XL,YO
90 DRAWTO (XR+XL)/2,YB
110 XR=XR+DXR:XL=XL+DXL
120 YB=YB+DYB:YO=YO+DYO
130 IF XL<=XR+10 OR YO<=YB THEN 130
140 GOTO 50
```

Voor een gek-makende kronkeltunnel kunt u de vorm laten kantelen:

```
10 GRAPHICS 8+16
20 X1=160:Y1=5
30 X2=30:Y2=185
40 X3=290:Y3=185
50 DX1=-1:DY1=3
60 DX2=3:DY2=-1
70 DX3=-3:DY3=-3
80 PLOT X1,Y1
90 DRAWTO X2,Y2:DRAWTO X3,Y3:DRAWTO X1,Y1
100 X1=X1+DX1:Y1=Y1+DY1
110 X2=X2+DX2:Y2=Y2+DY2
120 X3=X3+DX3:Y3=Y3+DY3
130 IF Y3<=Y1 THEN 130
140 GOTO 80
```

Als variatie op dit thema kunt u een regel laten vervallen en twee regels toevoegen. Haal regel 130 weg en voeg toe:

```
75 TRAP 150
150 GOTO 150
```

De rest van dit hoofdstuk zal gaan over de meer klassieke manier van het weergeven van diepte: het tekenen van de ribben nadat projectie plaats heeft gevonden. Vergeet bovenstaande methodes echter niet. Heeft u dit hoofdstuk uit, dan kunt u proberen door combinaties van technieken, nog mooiere tekeningen te maken.

Weergave op het beeldscherm

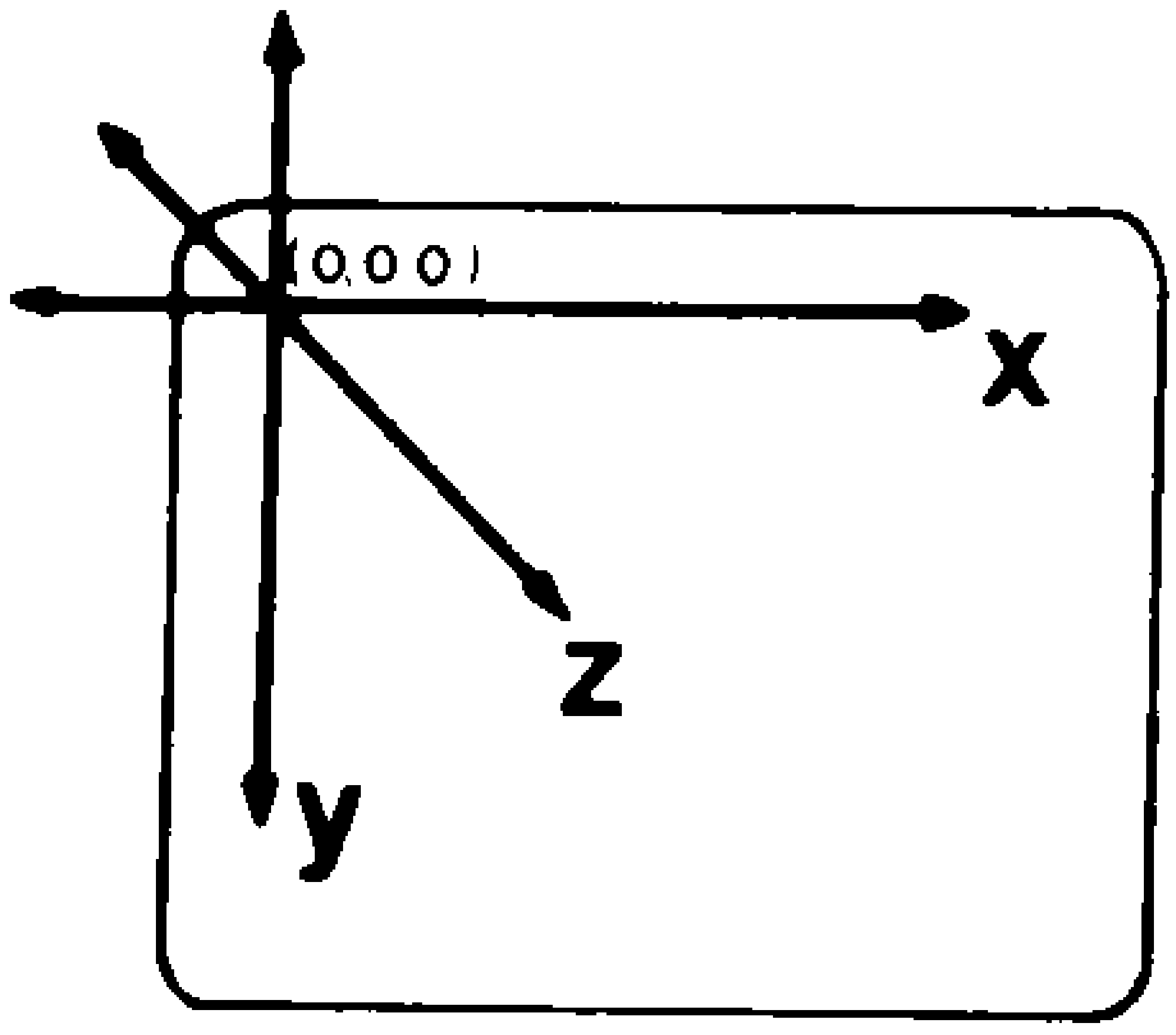
Het coordinatenstelsel om te tekenen in hoog oplossend vermogen bestaat uit een X-as (**links-rechts**) en een Y-as (**boven-beneden**). Door twee getallen op te geven, de X-coordinaat en de Y-coordinaat, kan een punt aangegeven worden. De notatie daarvoor is (X,Y).

De derde dimensie kan natuurlijk niet weergegeven worden op een beeldscherm. Deze moet er telkens even bijgedacht worden. In dit boek is gekozen voor een systeem van denken waarbij de kijker het scherm als het ware 'inkijkt'.

De X-as blijft waar hij is, van links naar rechts. De Y-as blijft waar hij is, van boven naar beneden. De diepte wordt aangegeven door een derde as, de Z-as. Deze Z-as gaat van het oppervlakte van het scherm de diepte in. Dat wil zeggen, van de kijker weg, de televisie in. De X-as staat loodrecht op de Y-as, maar ook loodrecht op de Z-as. De Z-as staat loodrecht op de X-as en op de Y-as.

Een punt wordt in 3D aangegeven door drie coordinaten (X,Y,Z). Het eerste getal geeft de verplaatsing links-rechts aan. Positieve getallen rechts van de oorsprong (het punt (0,0,0), negatieve getallen links van de oorsprong. Het tweede getal geeft de verplaatsing boven-beneden aan.

Positieve getallen onder de oorsprong, negatieve getallen erboven. Let op: dit is anders dan de normale wiskundige notatie. Het derde getal geeft de verplaatsing voor-achter (de diepte in) aan. Positieve getallen achter het oppervlak van het beeldscherm, van de kijker weg. Negatieve getallen voor het oppervlak van het beeldscherm, naar de kijker toe.



Assenstelsel

Matrices

Dit boek beoogt allerminst een wiskundeboek te zijn. Bij het projecteren van 3D objecten is een beetje matrix-rekenkunde echter onontbeerlijk. In dit boek vindt u in hoofdlijnen aangegeven hoe de, in de programma's gebruikte, formules tot stand zijn gekomen. Als u al veel kennis van matrix-rekenkunde heeft kan het een hulp zijn voor het verder uitwerken van experimenten. Voor hen die de matrix-rekenkunde niet machtig zijn, is het een basis om de werking van de programma's te kunnen doorgronden.

Voor de basis van de matrix-rekenkunde zoals die hier gebruikt wordt, moet u nog even het hiervoor behandelde 3D coordinatenstelsel in gedachten nemen. Elk punt in dit stelsel kan aangeduid worden met drie getallen; een X-, Y- en Z-coordinaat. Elk punt kan echter ook gezien worden als het resultaat van een stel verplaatsingen beginnend bij de oorsprong. Deze verplaatsingen worden **vectoren** genoemd. Zo is het punt (2,4,7) te beschouwen als het resultaat van een verplaatsing 2 maal een eenheid in de X-richting plus een verplaatsing van 4 maal een eenheid in de Y-richting plus een verplaatsing van 7 maal een eenheid in de Z-richting.
In een belangrijk deel van de wiskunde, de matrix-rekenkunde, worden meetkundige afbeeldingen (bijvoorbeeld rotaties) weergegeven met behulp van een matrix. Een matrix is een schema van een aantal getallen (of grootheden) die in de vorm van een rechthoek zijn geplaatst. Een matrix is te verdelen in rijen (horizontaal) en kolommen (vertikaal). Een voorbeeld van een meetkundige afbeelding die door een matrix wordt weergegeven, is:

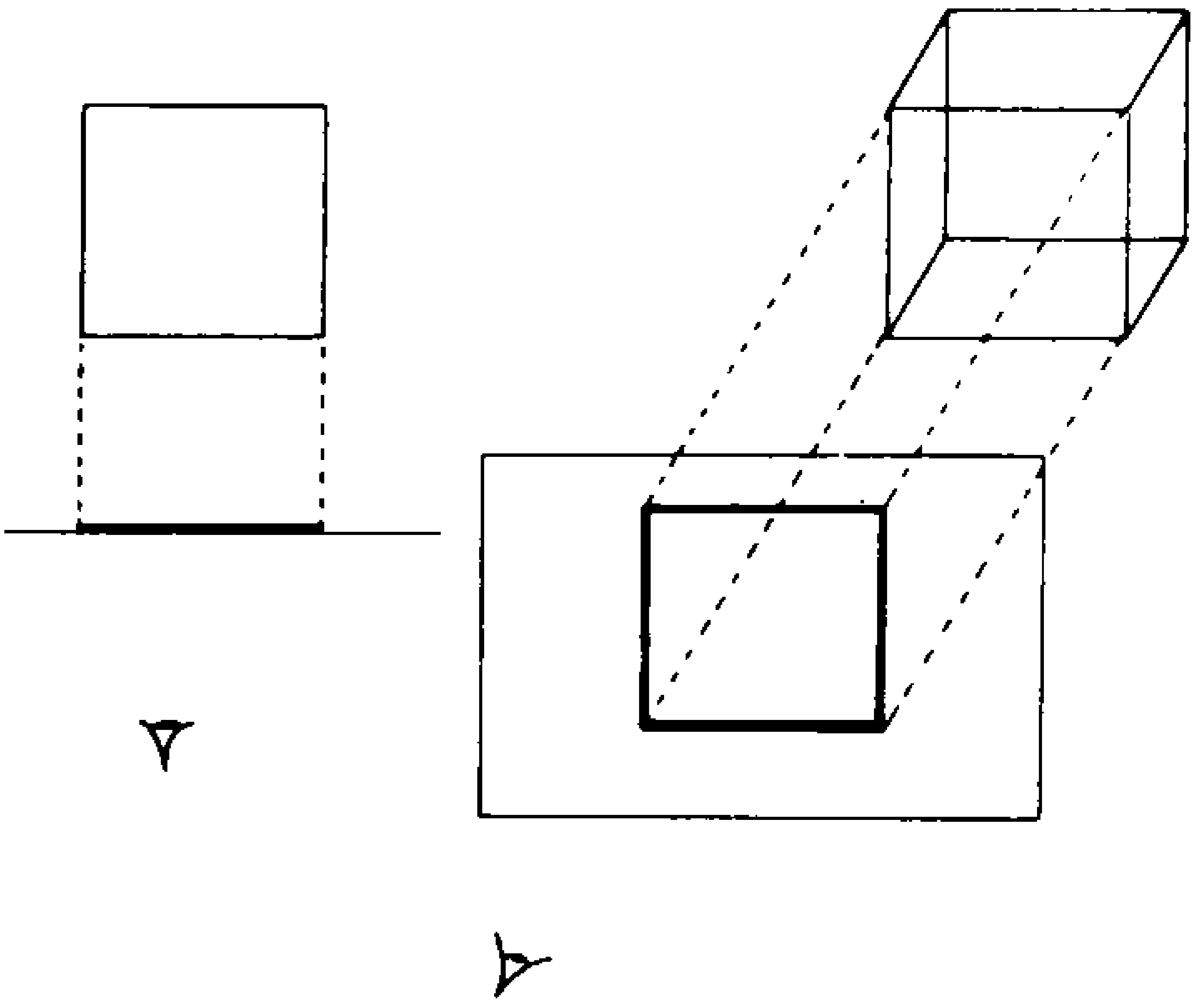
$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

In de eerste kolom van zo'n matrix staat genoteerd wat de meetkundige afbeelding doet met het punt (1,0,0). In ons voorbeeld wordt het punt (1,0,0)

veranderd in het punt (3,0,0). In de tweede kolom staat genoteerd wat er gebeurt met het punt (0,1,0). In ons voorbeeld wordt (0,1,0) veranderd tot (0,2,0). In de derde kolom staat genoteerd wat er gebeurt met het punt (0,0,1). Dat punt wordt (0,0,4).
Wat gebeurt er nu met het punt (2,4,7) als we deze meetkundige afbeelding toepassen?
(2,4,7) is te schrijven als 2*(1,0,0)+4*(0,1,0)+7*(0,0,1).
(2,4,7) wordt veranderd in 2*(3,0,0)+4*(0,2,0)+7*(0,0,4).
Dat wordt (6,0,0)+(0,8,0)+(0,0,28). Dat is gelijk aan het punt (6,8,28).
Deze eigenschappen van het matrixrekenen worden straks gebruikt bij de rotatie in een 3D ruimte.

Projectie

Zoals al vermeld is het onmogelijk om te tekenen binnen de televisie. Om een 3D object op het beeldscherm weer te geven moet er geprojecteerd worden. Eerst een voorbeeld van een eenvoudige projectie. Neem een vierkant en kijk recht van voren tegen de zijkant aan. U ziet dan alleen een lijnstuk, of neem een kubus en kijk recht van voren tegen een van de vlakken aan. U ziet dan een vierkant.



projectie van een vierkant en van een kubus

Het is echter niet de bedoeling om telkens recht tegen de voorkant van een object aan te kijken. Bij de hier gebruikte wijze van projecteren, de orthografische projectie, wordt van elk punt de Z-coordinaat nul gemaakt. Daardoor is na het projecteren van een kubus alleen een plat figuur te zien. De matrixnotatie voor het nulmaken van de Z-coordinaat is:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

De vector die zorgt voor de verplaatsing in de Z-richting wordt (0,0,0) gemaakt. De vectoren die zorgen voor de verplaatsing in de X- en in de Y-richting blijven onveranderd. Het punt (1,0,0) blijft (1,0,0). Het punt (0,1,0) blijft (0,1,0) en (0,0,1) wordt (0,0,0).

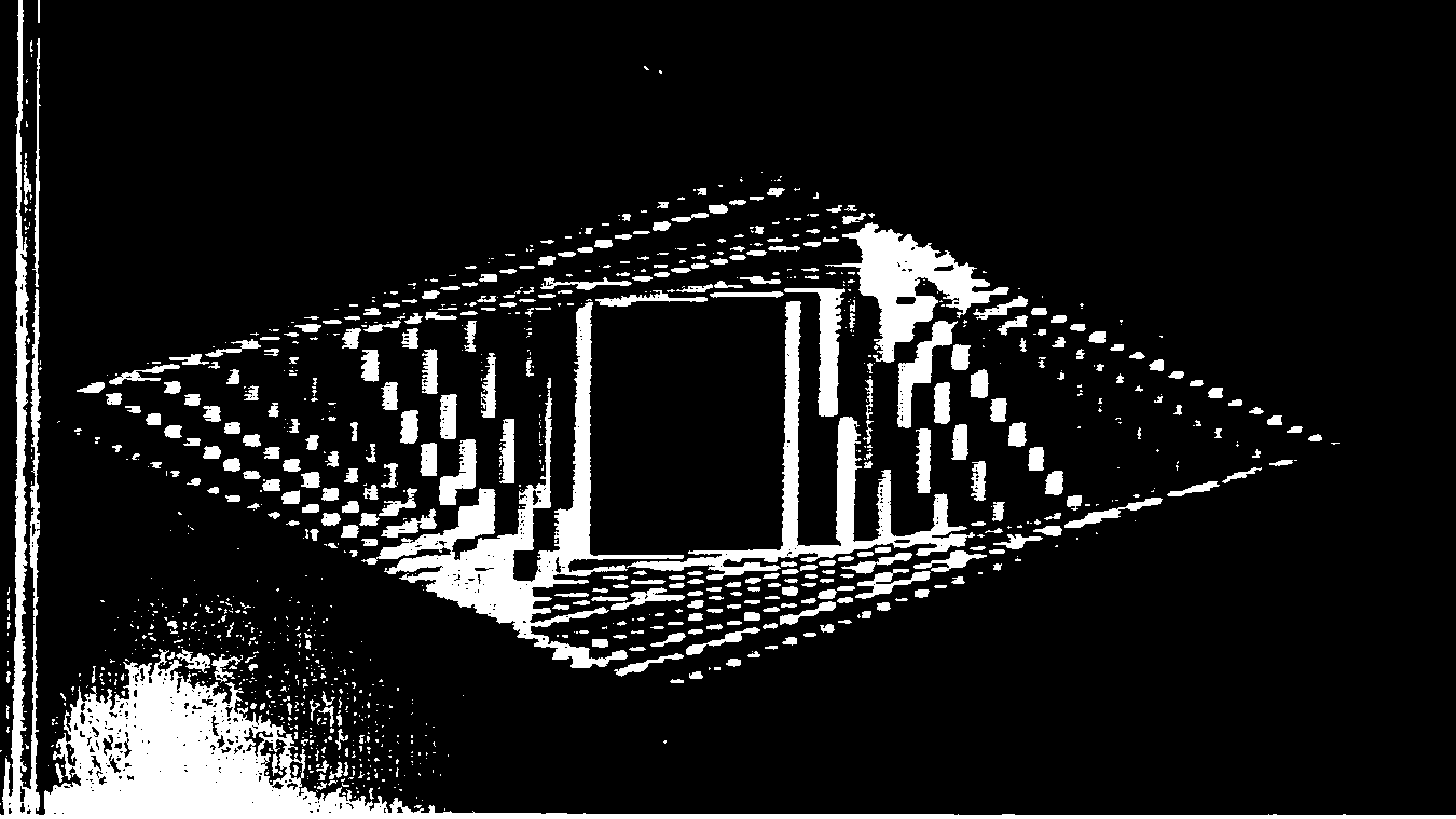
Als een bepaald punt moet worden geprojecteerd volgens de orthografische projectie kan de kolommatrix die het punt weergeeft (gelijk aan de notatie van de coordinaten, maar dan van boven naar beneden) vermenigvuldigd worden met de vierkante 3 bij 3 matrix, die staat voor deze vorm van projectie. Het punt (5,8,9) geeft na projectie het punt (5,8,0). Dat klopt, want alle punten moeten uiteindelijk in een vlak terecht komen (het vlak waarop [illegible] geprojecteerd).

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 5 & & 5 \\ 8 & = & 8 \\ 9 & & 0 \end{pmatrix}$$

Voor andere vormen van projectie (bijvoorbeeld de scheve projectie of de Mercatorprojectie) kunt u een wiskundeboek of een goede atlas naslaan.

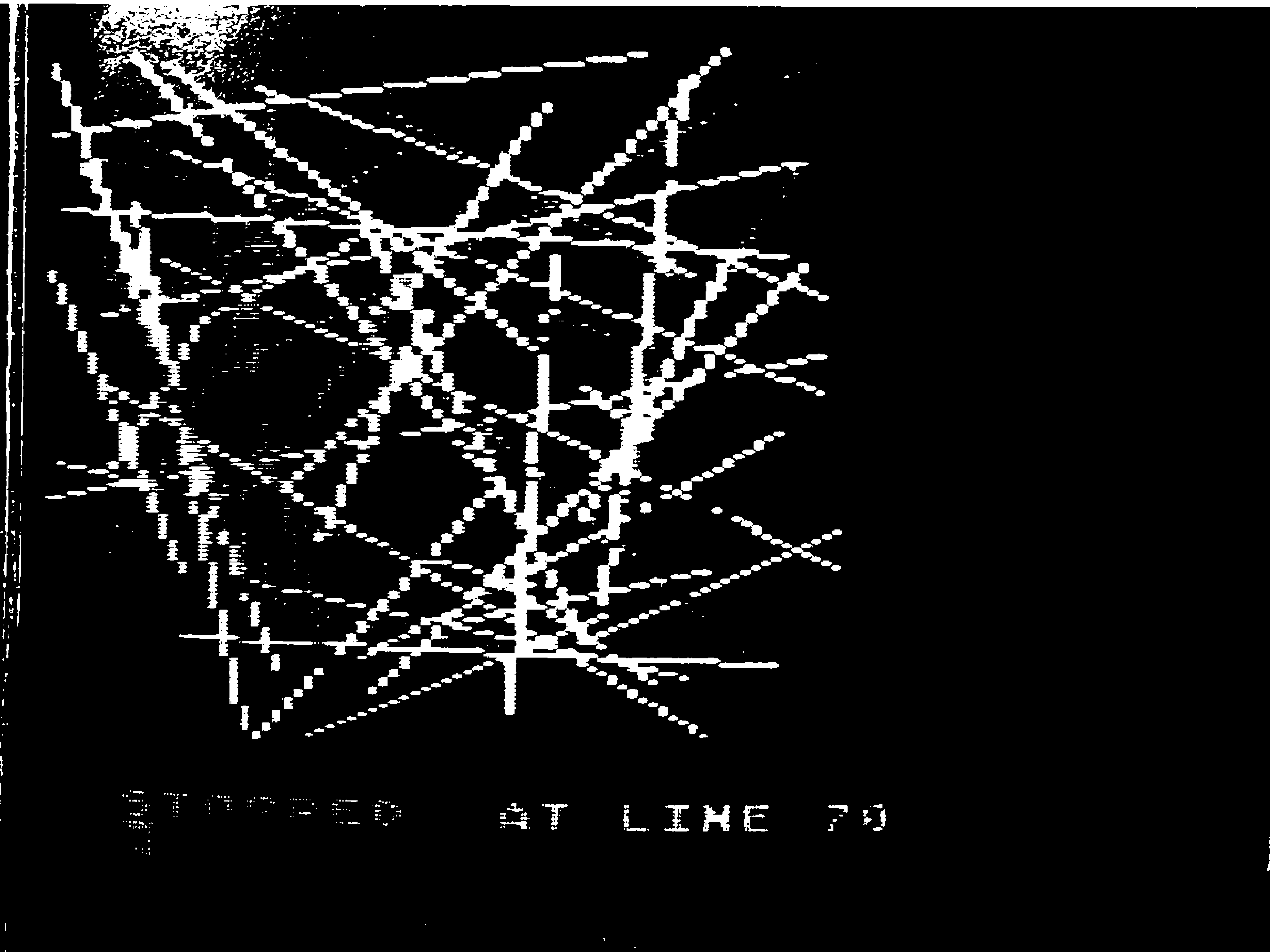
Veel mooier wordt een tekening als we schuin tegen een object aankijken. Dat is ook altijd de wijze waarop een kubus weergegeven wordt. De kubus wordt getekend vanuit een gezichtspunt iets opzij en iets boven de kubus. Om dit effect te bereiken zijn twee systemen mogelijk: De kijker verplaatst zich, of het object draait.

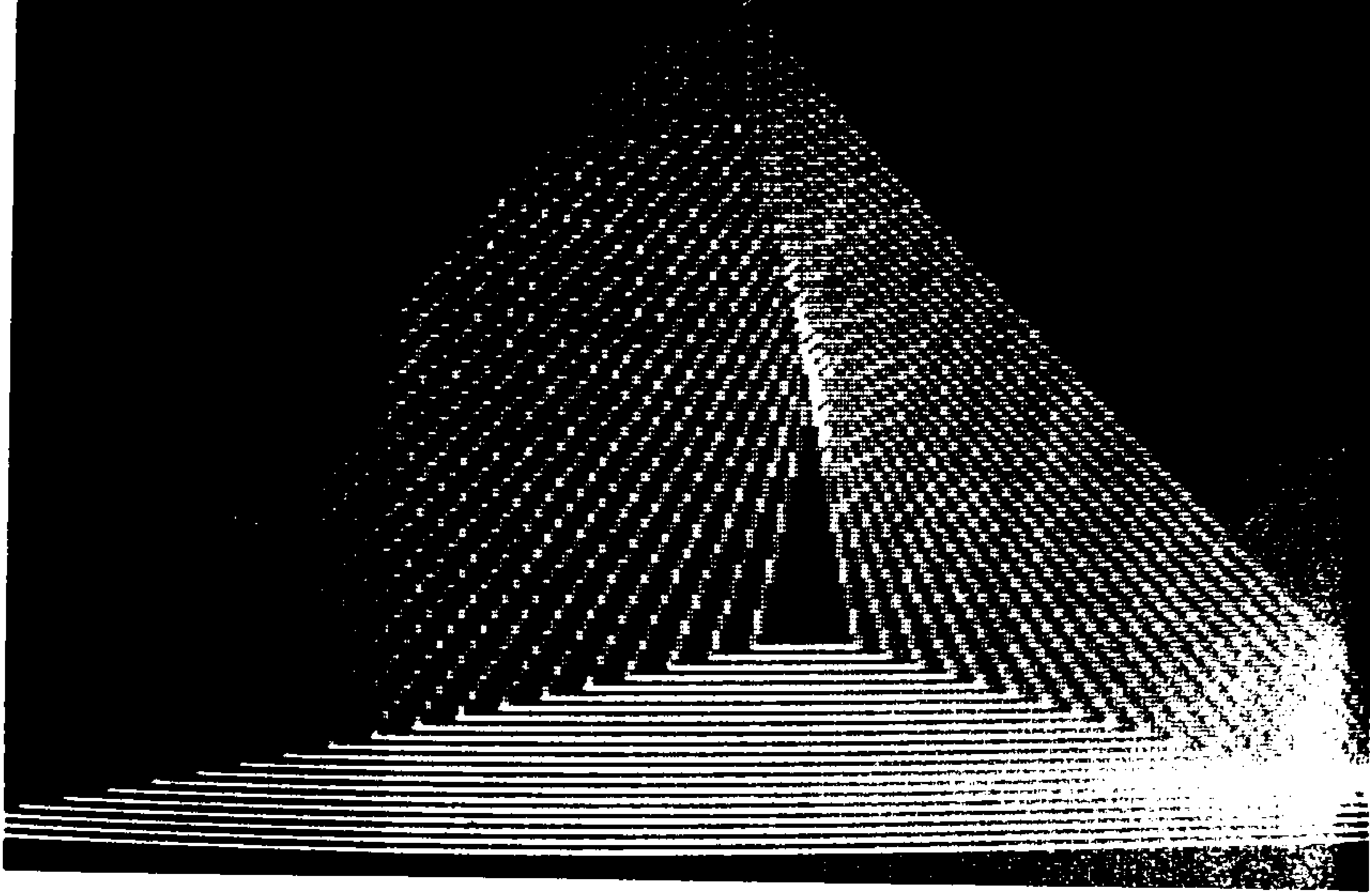
Er is geen wezenlijk verschil tussen deze twee, maar de tweede methode is wat eenvoudiger. ***Voor de wiskundige achtergrond van het volgende zij u verwezen naar middelbare-schoolboeken over vectormeetkunde, waarin de rotaties behandeld worden.*** Hier zal heel in het kort het principe uitgelegd worden.



Graphics voor soortgelijke programma's *hoofdstuk 14*

Graphics voor soortgelijke programma's *hoofdstuk 14*

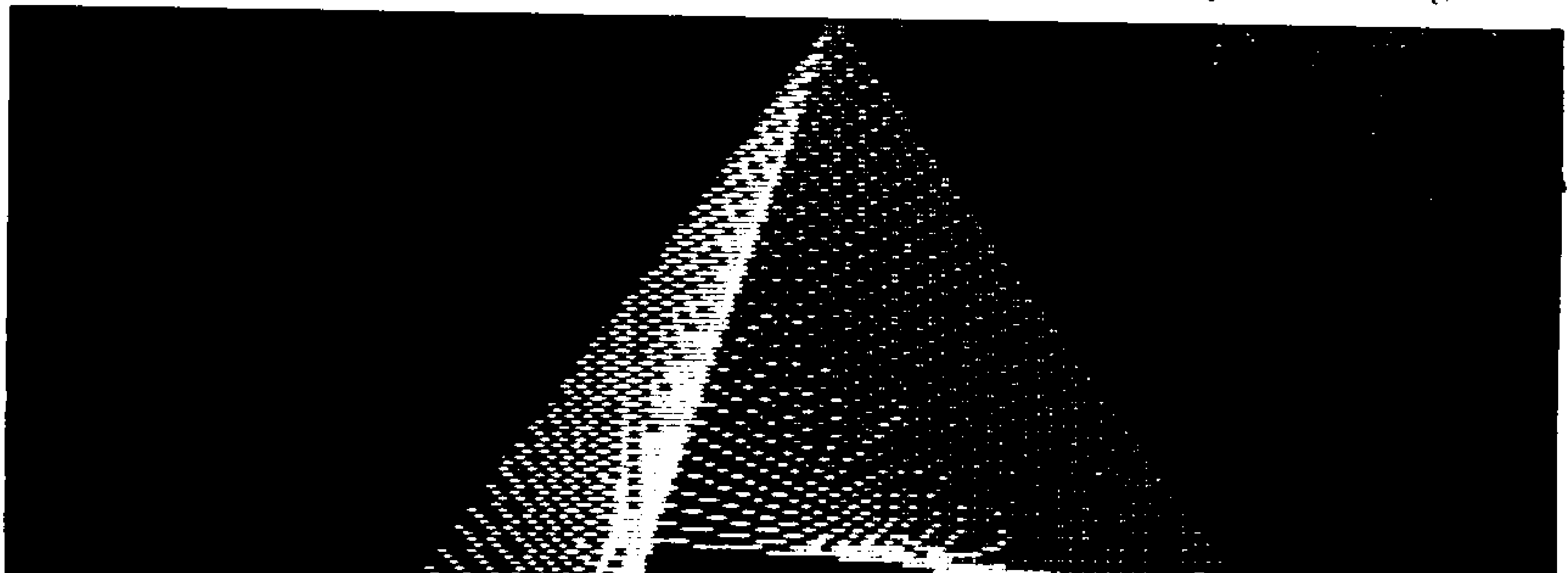




Tunnel programma *blz. 251*

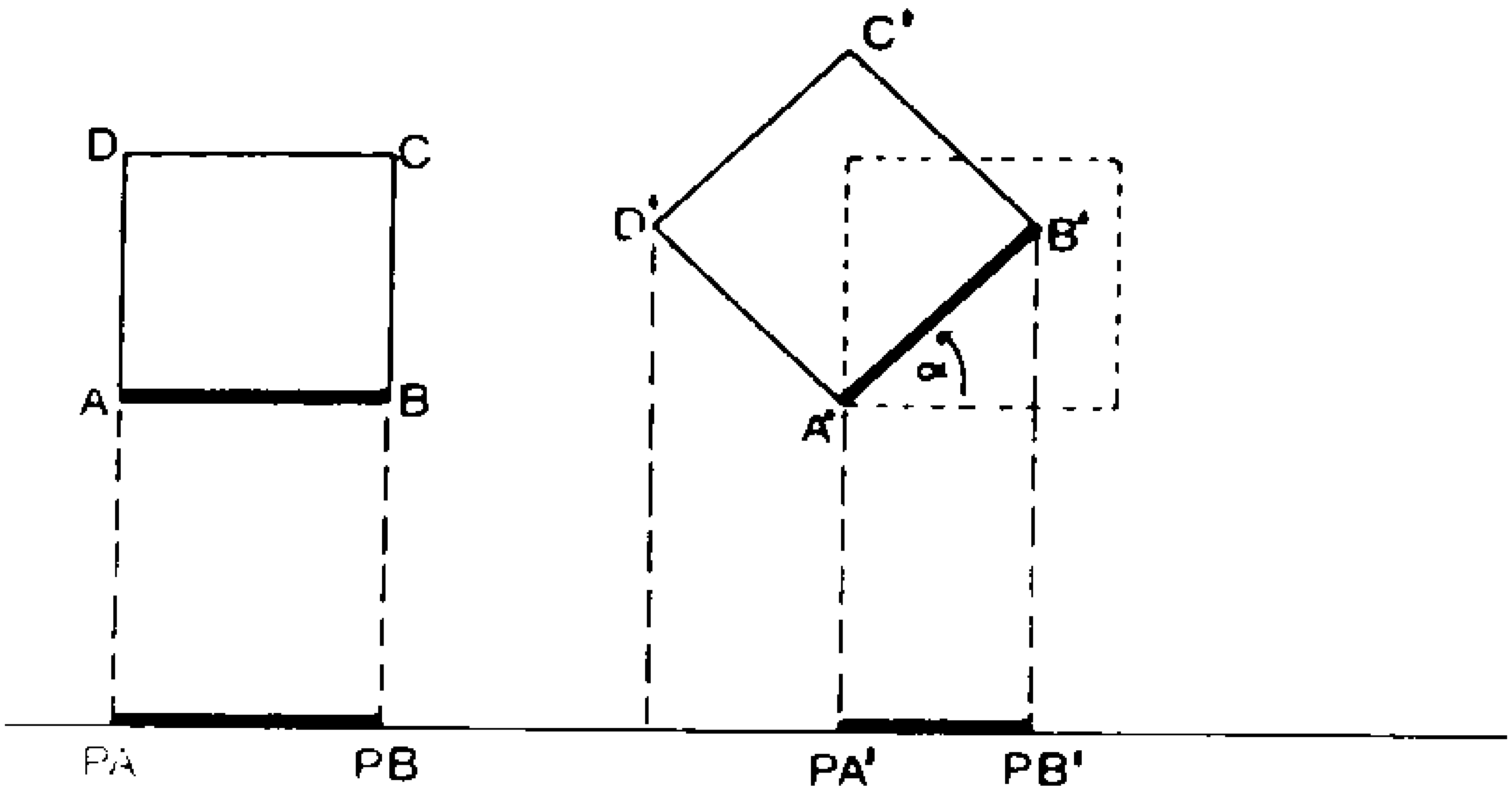
Tunnel "golvend" programma *blz. 251*

fotografie: Jorn van Engelen



Rotatie

Het vierkant uit het vorige figuur wordt gedraaid en daarna geprojecteerd op de X-as:



rotatie en projectie van een vierkant

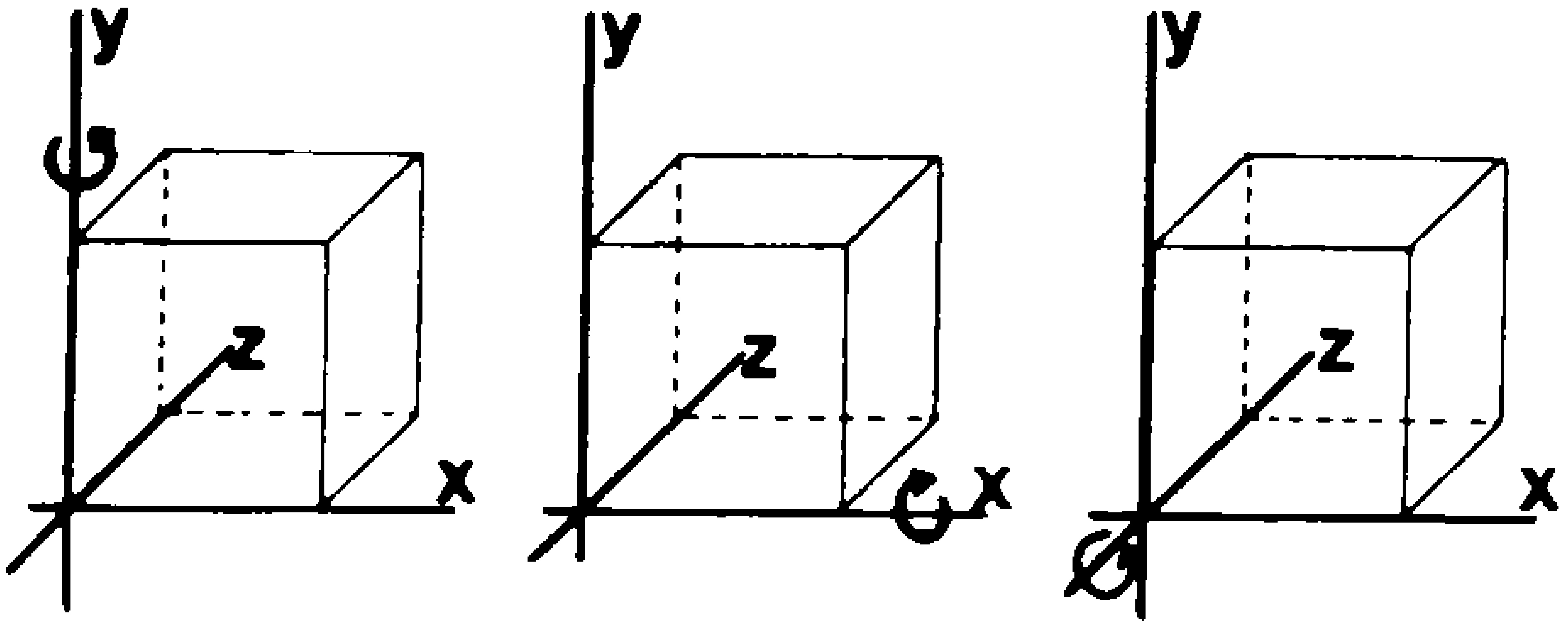
De punten A, B, C en D zijn de hoekpunten van het vierkant zonder dat het is gedraaid. De punten A', B', C' en D' zijn de hoekpunten van het vierkant na draaiing. De draaiing vond plaats over een hoek van 45 graden. De afstand AB is 1. Het figuur is een vierkant dus zijn alle andere zijden ook precies 1 eenheid lang.

Bij een projectie zonder draaiing is de afstand PA-PB (de afstand tussen de projectiepunten van A en B) gelijk aan 1. De afstand tussen PD en PC is ook 1. De afstand tussen PA en PD en tussen PB en PC is 0.

Na het draaien ziet het er heel anders uit. De afstand PA' en PB' is kleiner dan 1. Hoeveel kleiner dat is, is afhankelijk van de hoek waarover is gedraaid. Bij een draaiing over een hoek van 45 graden is de afstand tussen PA' en PB' ongeveer 0.7. Voor de wiskundige lezers: deze afstand is precies gelijk aan 0.5*SQR(2), oftewel COS(RAD(45)). Hoe meer er gedraaid wordt, hoe kleiner de afstand. Als er meer dan 90 graden wordt gedraaid, wordt de afstand zelfs negatief (PB' ligt dan links van PA').

Op deze wijze zijn de verschillende afstanden op de projectielijn uit te rekenen en zijn de verschillende projectiepunten te berekenen.

Rotaties in 3D gaan niet wezenlijk anders. Er is nu echter de keuze tussen drie verschillende rotaties: om de X-as, om de Y-as en om de Z-as.



verschillende rotatiemogelijkheden

De matrix van een rotatie over een hoek van A graden om de X-as ziet er als volgt uit:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & COS(A) & SIN(A) \\ 0 & -SIN(A) & COS(A) \end{pmatrix}$$

U ziet dat het punt (1,0,0) bij deze rotatie onveranderd blijft. Het punt (0,1,0) wordt veranderd in het punt (0,COS(A),-SIN(A)). Als we voor A bijvoorbeeld 30 graden nemen dan wordt het punt (0,1,0) veranderd in (0,0.866,-0.5) (0.866 is een benadering van 0.5*SQR(3), dat is de cosinus van 30 graden. 0.5 is de sinus van 30 graden).

De matrix van een rotatie over een hoek van B graden om de Y-as ziet er als volgt uit:

$$\begin{pmatrix} COS(B) & 0 & -SIN(B) \\ 0 & 1 & 0 \\ SIN(B) & 0 & COS(B) \end{pmatrix}$$

Bij deze rotatie blijft het punt (0,1,0) onveranderd.

De matrix voor een rotatie over een hoek van C graden om de Z-as is als volgt:

$$\begin{pmatrix} COS(C) & SIN(C)- & 0 \\ -SIN(C) & COS(C) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hier blijft het punt (0,0,1) onveranderd.

Een kubus

Het volgende programma tekent een kubus die niet recht van voren wordt bekeken. De kubus wordt eerst gedraaid om de X-as en daarna om de Y-as. De matrix die bij deze gecombineerde draaiing hoort, verkrijgt men door de matrix van een rotatie om de X-as te vermenigvuldigen met de matrix van een rotatie om de Y-as.
Een rotatie om de X-as over een hoek van A graden gevolgd door een rotatie om de Y-as over een hoek van B graden geeft:

$$\begin{pmatrix} COS(B) & 0 & -SIN(B) \\ 0 & 1 & 0 \\ SIN(B) & 0 & COS(B) \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & COS(A) & SIN(A) \\ 0 & -SIN(A) & COS(A) \end{pmatrix}$$

Het resultaat van deze vermenigvuldiging moet nog vermenigvuldigd worden met de projectiematrix (de matrix die ervoor zorgt dat de Z-coördinaten gelijk aan 0 worden). Vervolgens moet het resultaat vermenigvuldigd worden met de kolommatrix van het te roteren punt (X,Y,Z) van de kubus.

$$\begin{pmatrix} PX \\ PY \\ PZ \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} COS(B) & SIN(A)*SIN(B) & -COS(A)*SIN(B) \\ 0 & COS(A) & SIN(A) \\ SIN(B) & -SIN(A)*COS(B) & COS(A)*COS(B) \end{pmatrix} * \begin{pmatrix} X \\ Y \\ z \end{pmatrix}$$

Het eindresultaat:

PX = X*COS(B) + Y*SIN(A)*SIN(B) - Z*COS(A)*SIN(B)
PY = Y*COS(A) + Z*SIN(A)
PZ = 0

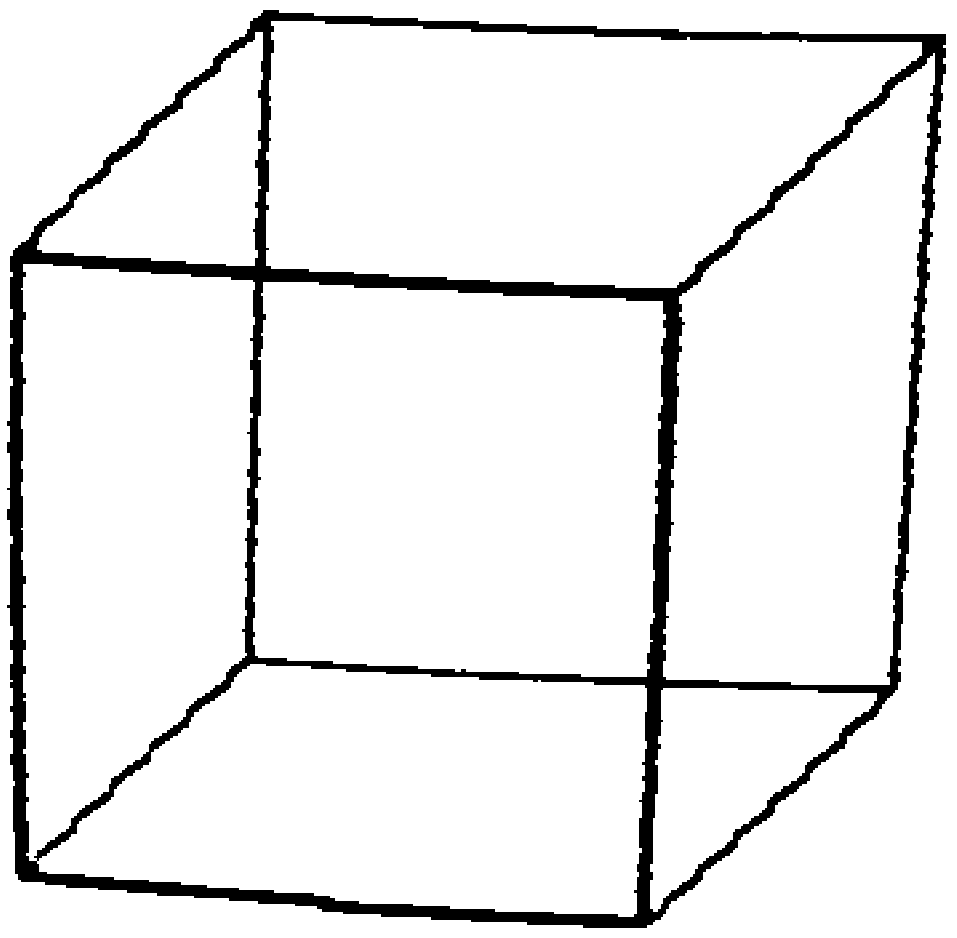
De waarden PX, PY en PZ staan voor de waarden van respectievelijk de X-coordinaat, de Y-coordinaat en de Z-coordinaat van het punt na projectie. Het punt heeft een Z-coordinaat die 0 is en ligt dus op het projectievlak (in ons geval het beeldscherm). Een rotatie die eerst om de Y-as gaat met een hoek van B graden en daarna om de X-as met een hoek van A graden en een projectie geeft als resultaat:

PX = X*COS(B) - Z*SIN(B)
PY = X*SIN(A)*SIN(B) + Y*COS(A) + Z*SIN(A)*COS(B)
PZ = 0

Dit programma laat een kubus in verschillende standen zien.

```
10 GRAPHICS 8:COLOR 1
20 DIM X(21),Y(21),Z(21)
30 FOR D=1 TO 21
40 READ A,B,C
50 X(D)=A:Y(D)=B:Z(D)=C
60 NEXT D
```

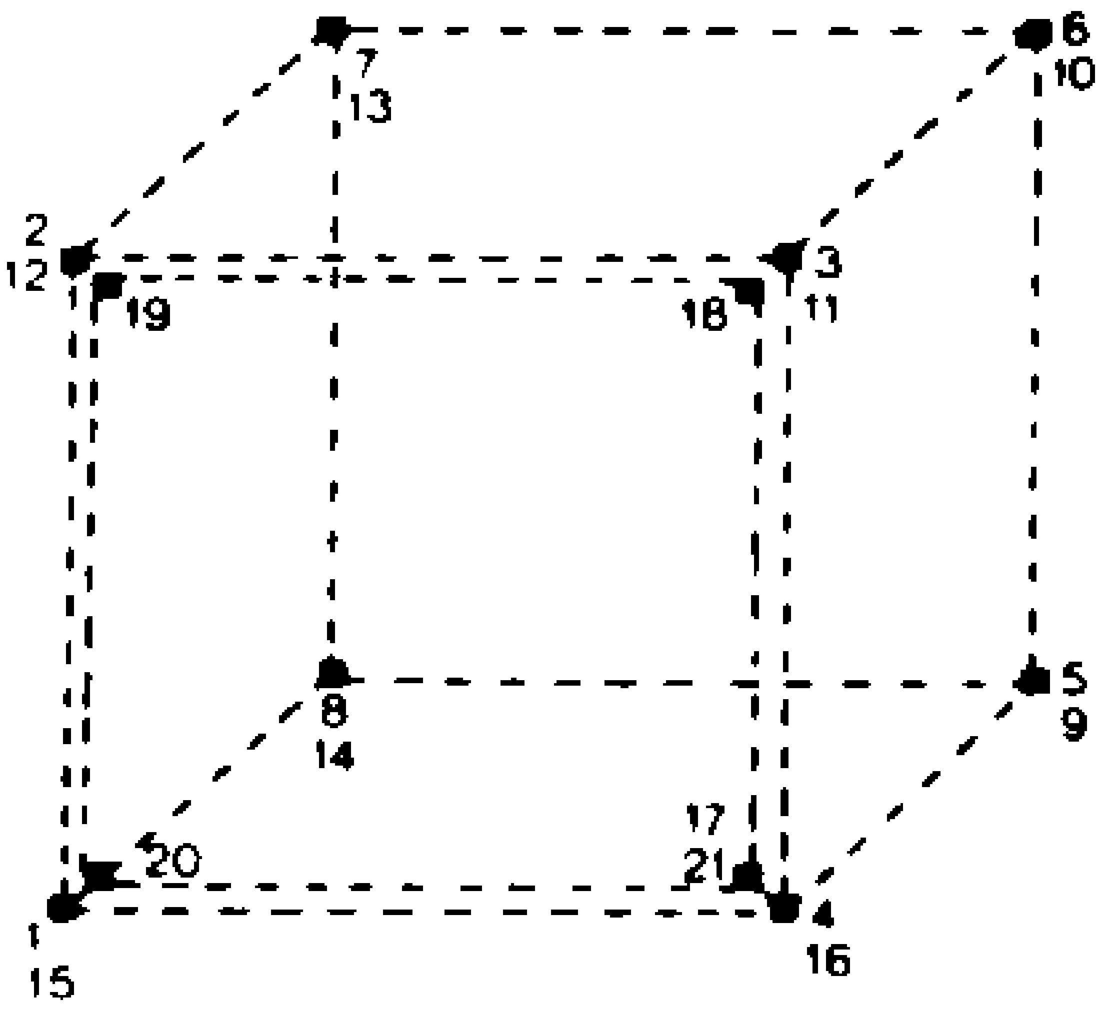
```
70 PRINT "DRAAIING OM Y-AS";:INPUT A
80 PRINT "DRAAIING OM X-AS";:INPUT B
85 TRAP 250
90 A=(A*3.14)/180
100 B=(B*3.14)/180
110 PLOT 100,130
120 K=COS(A):L=SIN(A)*SIN(B):M=SIN(A)*COS(B):N=COS(B):J=SIN(
130 FOR D=1 TO 21
140 PX=X(D)*K+Y(D)*L-Z(D)*M
150 PY=Y(D)*N+Z(D)*J
160 DRAWTO PX+100,130-PY
170 NEXT D
180 FOR WA=1 TO 3000:NEXT WA
190 GRAPHICS 8
200 GOTO 70
210 END
250 PRINT "KUBUS KOMT BUITEN BEELD."
260 PRINT "VUL NIEUWE WAARDEN IN S.V.P."
270 GRAPHICS 8:GOTO 70
300 DATA 80,0,0,80,80,0,0,80,0,0,0,0
305 DATA 1,1,1,79,1,0
310 DATA 79,79,0,1,79,0,1,1
315 DATA 0,80,0,0,80,0,80
320 DATA 80,80,80,80,80,0,80,80,80
330 DATA 0,80,80,0,80,0,0,0,0,0,0,80
340 DATA 80,0,80,0,0,80,0,80,80
```



een kubus getekend door bovenstaand programma

In regel 20 wordt voldoende plaats gemaakt voor het tekenen van de kubus. In de regels 30-60 worden de verschillende gegevens voor de kubus in een array geplaatst. Daardoor gaat het tekenen sneller.
De gegevens die in de datalijst zijn opgenomen, zijn de punten waarnaar een lijn al dan niet getrokken moet worden (telkens een X-, Y- en Z-coordinaat). In het voorbeeldprogramma kon volstaan worden met het telkens trekken van een lijn. Een kubus kan worden getekend door 16 lijnen te trekken. Om te

zorgen dat de voorste lijnen ietsje dikker worden (hetgeen het 3D effect vergroot) zijn 21 lijnen nodig volgens onderstaand schema:



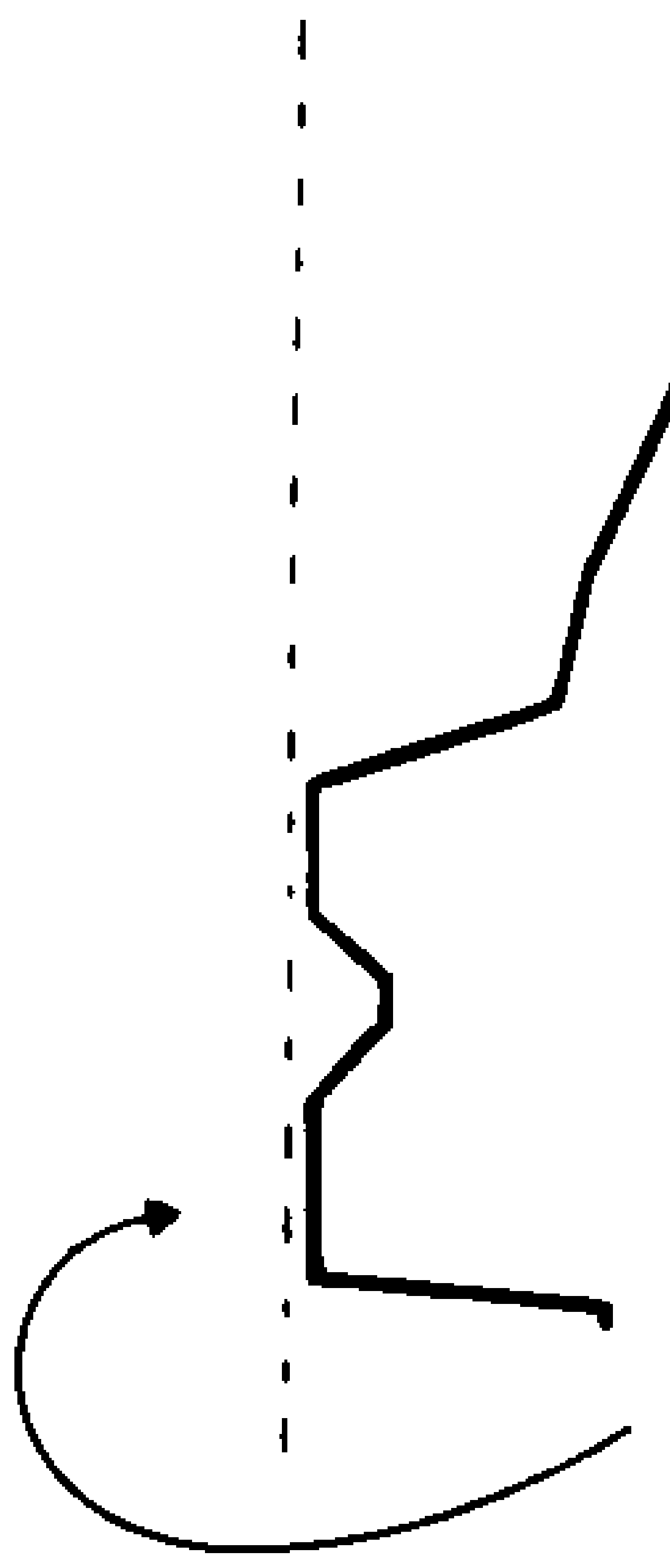
Als u andere figuren dan kubussen wilt tekenen, zult u soms te maken krijgen met tekeningen waarbij het veel eenvoudiger is om 'het potlood' even te verplaatsen zonder een lijn te trekken. U moet dan elk coordinatenstel voorzien van een vierde getal (0 of 1) voor het al dan niet tekenen van de lijn. Het is het handigste om daarbij altijd 0 aan te geven als er geen lijn moet worden getrokken en 'het potlood' alleen moet worden verplaatst. Een 1 staat dan voor het trekken van een lijn. Een eenvoudige IF..THEN..opdracht zorgt ervoor dat op de juiste plaatsen de lijn wordt getrokken.
De regels 70 en 80 vragen de gebruiker om welke hoek de vorm gedraaid moet worden voordat hij geprojecteerd en getekend wordt.
De regels 90 en 100 zijn wat cryptisch. De meest gebruikte wijze om hoeken aan te geven is in graden. Er gaan 360 graden in een volle cirkel. Om een programma een beetje gebruikersvriendelijk te maken is het in elk geval nodig dat de gebruiker de hoeken in een voor hem/haar bekende grootheid op kan geven. In dit geval graden. De Atari werkt echter met radialen. In een volle cirkel gaan 2*PI radialen. PI is ongeveer gelijk aan 3.14. In de regels 90 en 100 rekent de Atari zelf de hoek in graden om in radialen. Dit is de tweede methode om het verschil tussen graden en radialen te overkomen. De andere methode vindt u in de paragraaf over dieptesuggestie door schaduwwerking.
Regel 110 verplaatst de grafische cursor naar de beginpositie.
Regel 120 moet u in samenhang met de regels 140 en 150 lezen. De laatste twee regels worden voor elk punt herhaald. Regel 120 wordt maar een keer per tekening uitgevoerd. Door op deze wijze een deel van de berekeningen naar een programmaregel te verplaatsen, waar zij maar een keer uitgevoerd hoeven te worden in plaats van vele keren, kan het tekenen vele malen sneller gaan. ***De berekening die de computer bij elk te tekenen punt moet maken is nu veel korter.*** Test u dit uit door in regel 140 en 150 de oorspronkelijke berekening te plaatsen:

```
140 PX=X(D)*COS(A)+Y(D)*SIN(A)*SIN(B)-Z(D)*SIN(A)*COS(B)
150 PY=Y(D)*COS(B)+Z(D)*SIN(B)
```

Ziet u hoeveel langzamer het tekenen gaat? Bij het maken van uw eigen tekenprogramma's waar veel rekenwerk in voorkomt moet u hier goed op letten: laat zo veel mogelijk rekenwerk van te voren (dat wil zeggen buiten de tekenlus) doen. Een herhalingslus moet zo kort mogelijk zijn om snelle tekeningen te kunnen maken.
Regel 160 zorgt voor het uiteindelijke tekenwerk. Regel 160 zou eigenlijk overbodig moeten zijn. Door de coordinaten zorgvuldig te kiezen kan ervoor worden verzorgd dat een verschuiving niet meer nodig is. Anderzijds heeft een verschuivingsfactor het voordeel dat de vorm over het scherm heen en weer kan worden bewogen.
Regel 180 wacht een tijdje voordat regel 190 weer terugschakelt naar scherm 8, zodat gebruiker een nieuw stel hoeken op kan geven.

Een wijnglas

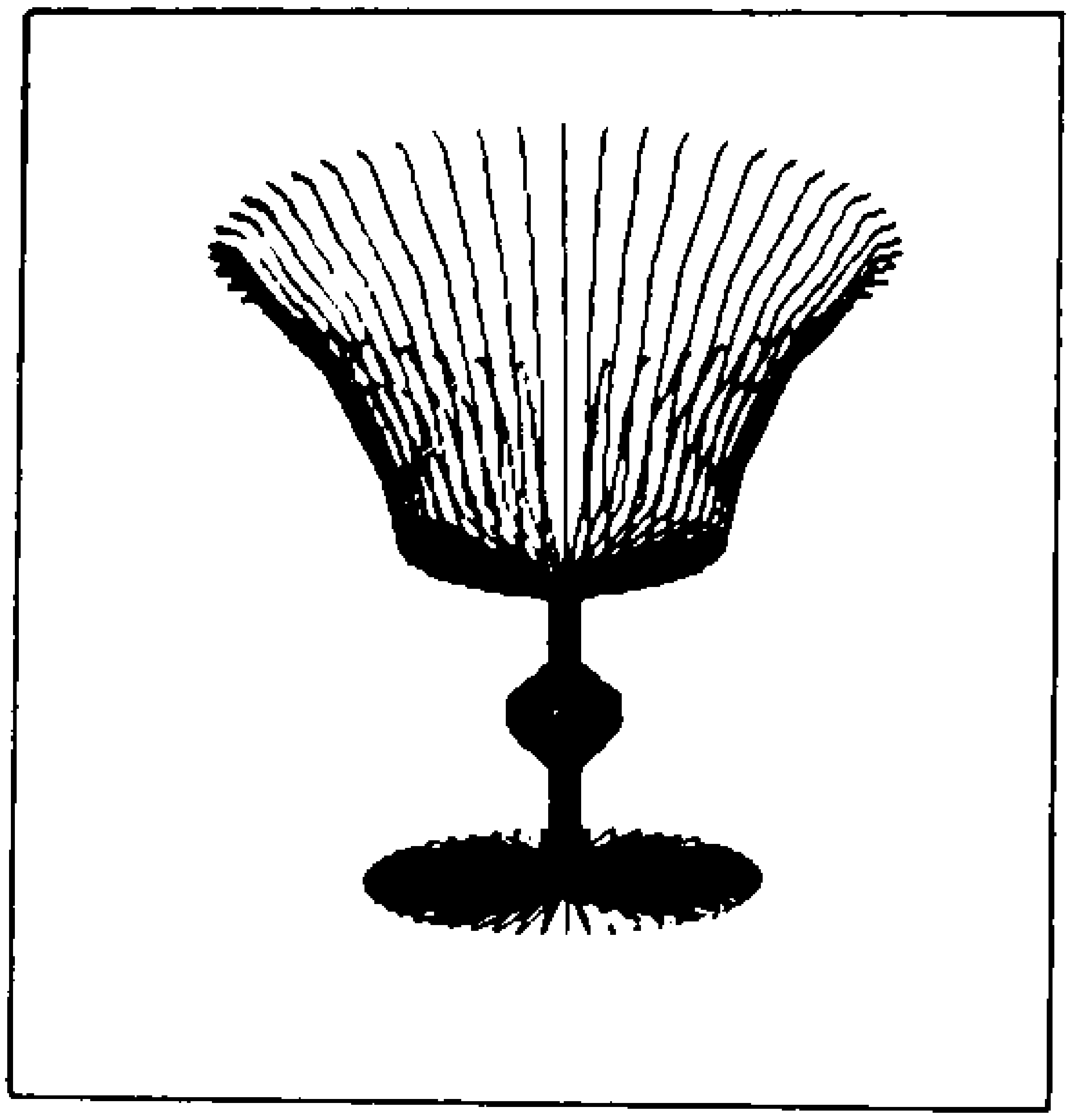
Door het vorige programma wat uit te breiden kunt u het uzelf nog gemakkelijker maken. Het onderstaande programma tekent niet alleen een door u bedacht voorwerp, het bedenkt zelfs hoe het voorwerp er uit moet zien. Het enige dat u hoeft in te voeren is een willekeurige gekromde lijn. De computer draait deze lijn telkens een paar graden rond. De rotatie vindt plaats rond de Y-as.



Na 360 graden is de lijn weer op zijn oude standpunt.
Omdat de kijker tijdens dit proces recht van voren tegen de draaiende lijn

aankijkt, lijkt het of de computer een eenvoudige 2D tekening heeft gemaakt. Maar de computer heeft tijdens het draaien de verkregen gegevens opgeslagen in een array. Als u na het vormen van het figuur de computer vraagt het voorwerp te roteren rond de X-as kunt u het voorwerp van verschillende kanten bekijken.

```
10 GRAPHICS 8+16
20 SETCOLOR 2,0,0:COLOR 1
30 DIM X(9),Y(9),Z1(33,9),Z2(33,9)
40 FOR Q=1 TO 9
50 READ A,B
60 X(Q)=A:Y(Q)=B
70 NEXT Q
80 FOR D=1 TO 32
90 PLOT 125,170
100 S=SIN(D*0.2)
110 C=COS(D*0.2)
120 FOR A=1 TO 9
130 Z1(D,A)=S*X(A)
140 Z2(D,A)=C*X(A)
150 DRAWTO Z1(D,A)+125,Y(A)+160
160 NEXT A
170 PLOT 125,160
180 NEXT D
190 FOR B=0.5 TO 0.8 STEP 0.3
200 GRAPHICS 8+16:SETCOLOR 2,0,0
210 PLOT 125,160
220 N=COS(B):J=SIN(B)
230 FOR D=1 TO 32
240 PLOT 125,160
250 FOR A=1 TO 9
260 PX=Z1(D,A)
270 PY=Y(A)*N+Z2(D,A)*J
280 DRAWTO PX+125,PY+160
290 NEXT A:NEXT D
295 FOR WA=1 TO 3000:NEXT WA
300 NEXT B
310 GOTO 310
500 DATA 40,0,4,6,4,-30,10,-45,4,-55,4
510 DATA -70,30,-80,35,-106,55,-150
```



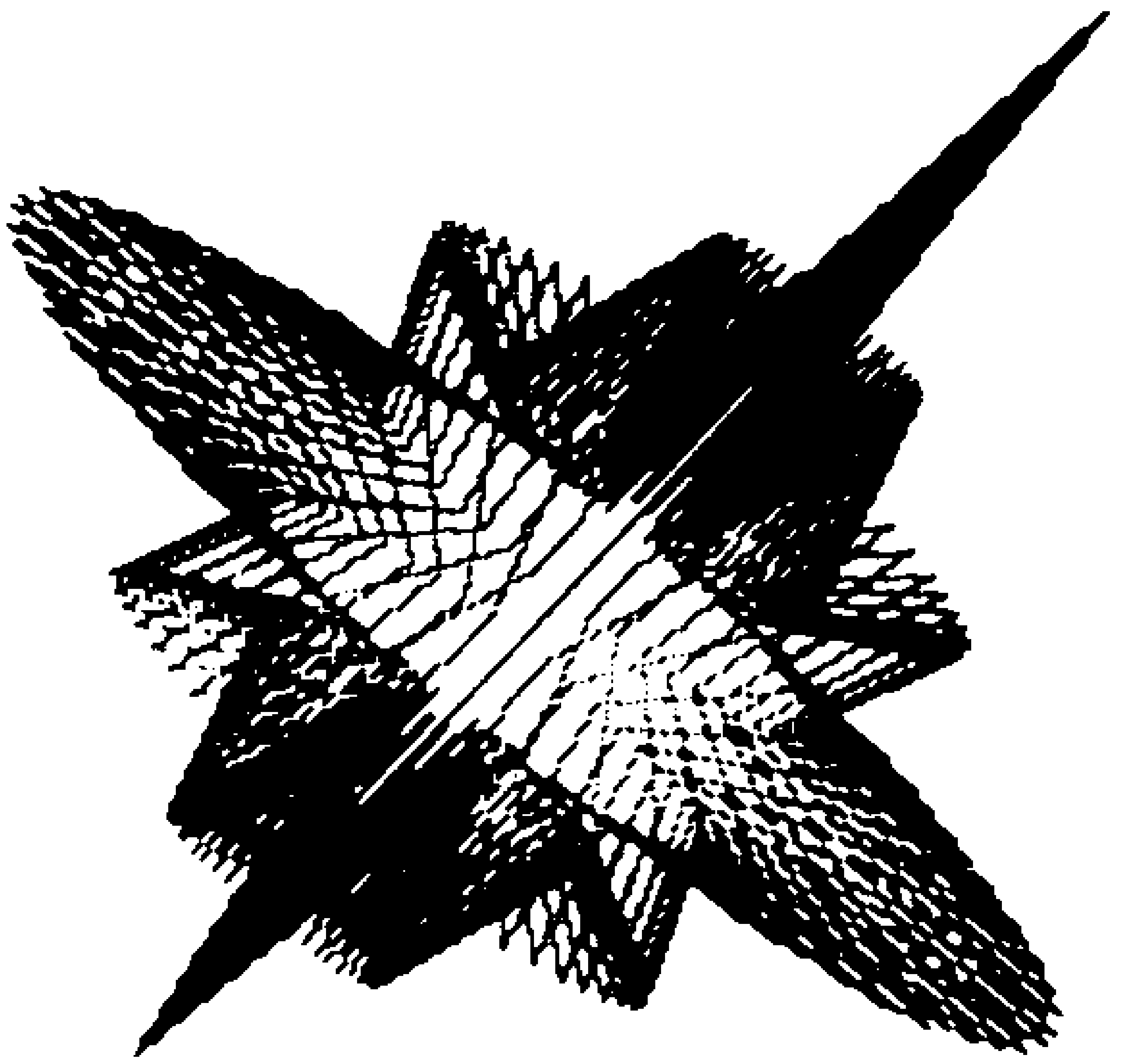
Dit programma laat als voorbeeld een wijnglas zien. Nadat het glas eerst in 2D is getekend, waarbij de gegevens voor de 3D weergave gelijk worden berekend, wordt vervolgens het glas in verschillende 3D standen getoond. De hoeken waarover het glas om de X-as geroteerd wordt, kunt u (in radialen) aflezen in regel 190. Roteren rond de Y-as heeft niet zoveel zin, omdat het voorwerp verkregen is door rotatie rond de Y-as en perfect symetrisch is. Door de draaiing rond de X-as kunt u het glas van binnen bekijken.
Regel 30 reserveert ruimte voor de 2D en 3D weergave.
De regels 40-70 lezen de gegevens uit de datalijst. De gegevens bestaan uit telkens twee getallen die een punt van de hiervoor getekende lijn aangeven.
De regels 80-180 tekenen 32 keer alle negen punten van de lijn. Telkens wordt de lijn een beetje gedraaid. In de 2D weergave is dat te zien doordat de lijn verschuift.
De regels 130 en 140 plaatsen de 3D gegevens in de array.
Regel 150 zorgt voor het tekenen van de 2D uitvoering.
Regel 170 zorgt voor het verplaatsen van de grafische cursor.
Vanaf regel 190 wordt de 3D weergave verzorgd. Heeft u liever dat u zelf de hoek waarover het wijnglas draait kunt opgeven, dan kunt u regel 190, 200 en 300 veranderen zodat er een INPUT opdracht komt te staan (daarvoor moet u tijdelijk naar een schermmodus 0 overschakelen, al dan niet als tekstvenster).
De regels 210-290 zijn vergelijkbaar met de regels 80-180. Nu hoeft er echter niets uitgerekend te worden, maar kan de computer direkt de gegevens uit de array gebruiken om te gaan tekenen. In regel 220 wordt vast een deel van het rekenwerk gedaan. Dit is eens stuk eenvoudiger dan in het kubusprogramma. Dat komt doordat er slechts over een as gedraaid hoeft te worden.

Om in plaats van een wijnglas een ander figuur te krijgen hoeft u alleen de gegevens te veranderen. Let u hierbij wel op: als u meer gegevens plaatst, moet het aantal keren dat een gegeven wordt gelezen, aangepast worden (regels 40,

120 en 250). Tevens moet de dimensionering van de arrays eventueel worden aangepast (regel 30).
Verander in het vorige programma de volgende regels:

```
 30 DIM X(13),Y(13),Z1(33,13),Z2(33,13)
 40 FOR Q=1 TO 13 -120 FOR A=1 TO 13
250 FOR A=1 TO 13
500 DATA    0,10,7,-30,15,0,22,-30,40,-10,    30,-40,75,-50,30,-60,40,-
    90,22,-70,15, -100,7,-70,0,-150
```

Voor andere voorwerpen kunt u de gegevens zelf veranderen. De gegevens horen twee aan twee bij elkaar. Telkens een X-coordinaat en een Y-coordinaat van een punt van de gekromde lijn die moet worden gedraaid. Let op de ruimtereservering voor de verschillende arrays en het aantal keren dat de lussen moeten worden doorlopen.



U kunt het uzelf nog gemakkelijker maken door zelfs de gekromde lijn niet punt voor punt op te geven, maar de computer te vragen om aan de hand van een door u opgegeven formule de punten te bepalen. Op deze wijze krijgt u mooie ribbelvlakken. Vervang, van het al volgens bovenstaande aanwijzingen veranderde programma, de volgende regels:

```
 50 Y(Q)=-50-70:SIN(Q-9.9)/Q):X(Q)=Q*7+10
210 PLOT 125,130
240 PLOT 125,130
280 DRAWTO PX+125,PY+130
```

De datalijst heeft u nu niet meer nodig.
Probeer ook:

```
50 Y(Q)=-60-50*(SIN(((Q/2.5)-.7)/Q-.9)) : X(Q)=Q*5+30
```

voor een knikkerputje.
Heel mooi is de volgende formule:

```
70 Y(Q)=-110-.5*(Q*Q)*(SIN(Q)/Q-.9) : X(Q)=7*Q+10
```

Een kubus

Het programma hieronder laat zien hoe een object rond alle drie de assen kan worden gedraaid. Dit programma is opvallend door de snelheid. Dit ondanks de vele berekeningen die voor elk punt opnieuw moeten worden gemaakt. Bij een draaiing om drie assen worden de berekeningen die gemaakt moeten worden gemaakt, om het projectiepunt van elk punt te bepalen, een stuk langer. In dit programma wordt altijd eerst gedraaid om de X-as, daarna om de Y-as en tot slot rond de Z-as. Deze volgorde kan worden veranderd, maar is bepalend voor voor het uiteindelijke resultaat. Door echter de juiste draaiingshoeken te kiezen kan de gebruiker elke stand van het object verkrijgen.
De projectieformule is op de volgende wijze gevonden:
De matrix van een projectie op het X-Y vlak (Z=0) na een rotatie om de Z-as, na een rotatie om de Y-as, na een rotatie om de X-as is berekend. Deze matrix is gevonden door eerst de matrices voor rotatie om de Y-as en de X-as te vermenigvuldigen. Vervolgens wordt het resultaat hiervan vermenigvuldigd met een matrix voor een rotatie om de Z-as (over hoek C). Het resultaat is:

$$\begin{pmatrix} \text{COS(B)*COS(C)} & \text{SIN(A)*SIN(B)*COS(C)} & \text{-COS(A)*SIN(B)*COS(C)+} \\ & \text{+COS(A)*SIN(C)} & \text{SIN(A)*SIN(C)} \\ \text{-COS(B)*SIN(C)} & \text{-SIN(A)*SIN(B)*SIN(C)+} & \text{COS(A)*SIN(B)*SIN(C)+} \\ & \text{COS(A)*COS(C)} & \text{SIN(A)*COS(B)} \\ \text{SIN(B)} & \text{-SIN(A)*COS(B)} & \text{COS(A)*COS(B)} \end{pmatrix}$$

Deze matrix wordt vermenigvuldigd met de projectiematrix (Z=0).

De gebruiker kan in het volgende programma kiezen uit twee mogelijkheden: datalijst of data-invoer. Kiest u voor de datalijst dan hoeft u verder niets meer te doen. Het programma heeft een stel gegevens klaar om een kubus te tekenen waarmee getoond kan worden wat het programma allemaal kan. Wilt u een ander voorwerp, dan kan dat door de gegevens in te voeren na het kiezen van data-invoer. De gegevens die u dan invoert blijven echter niet in het programma staan. Als u veel verschillende voorwerpen met veel gegevens wilt gebruiken bij dit of een van de volgende programma's dan kunt u deze gegevens het

beste opslaan in aparte bestanden. Lees hiervoor het hoofdstuk over het werken met bestanden.

```
10 GRAPHICS 0
20 PRINT "DATALIJST (0) OF DATAINVOER (1)";
25 INPUT E
30 IF E=1 THEN 140
40 RESTORE
50 READ L
60 DIM X(L),Y(L),Z(L),T(L)
70 FOR H=1 TO L
80 READ A,B,C,D
90 X(H)=A:Y(H)=B
100 Z(H)=C:T(H)=D
120 NEXT H
130 GOTO 250
140 PRINT "AANTAL PUNTEN";:INPUT L
150 DIM X(L),Y(L),Z(L),T(L)
160 FOR H=1 TO L
170 PRINT "X-COORD. PUNT ";H;" ";:INPUT A
180 X(H)=A
190 PRINT "Y-COORD. PUNT ";H;" ";:INPUT B
200 Y(H)=B
210 PRINT "Z-COORD. PUNT ";H;" ";:INPUT C
220 Z(H)=C
230 PRINT "LIJN (1) OF VERPLAATS (0)";
235 INPUT D:T(H)=D
240 NEXT H
250 GRAPHICS 8+32
260 PRINT "   INVOER IN GRADEN"
270 PRINT "ROTATIE OM X-AS ";:INPUT A1
280 PRINT "ROTATIE OM Y-AS ";:INPUT B1
290 PRINT "ROTATIE OM Z-AS ";:INPUT C1
300 GRAPHICS 8+16
310 GOSUB 540:REM HOEKBEREKENING
320 GOSUB 420:REM TEKENEN
330 FOR WA=1 TO 3000:NEXT WA
340 GOTO 250
350 END
360 DATA 21
370 DATA 0,0,0,0,0,50,0,1,50,50,0,1,50
375 DATA 0,0,1,50,0,50,1,50,50,50,1
380 DATA 0,50,50,1,0,0,50,1,50,0,50,1
385 DATA 50,50,50,0,50,50,0,1,0,50,0,0
390 DATA 0,50,50,1,0,0,50,0,0,0,0,1,50
395 DATA 0,0,1,48,2,0,1,48,48,0,1,2,48,0,1
```

```
400 DATA 2,2,0,1,48,2,0,1
410 REM TEKENEN
420 PLOT 125,125
430 FOR H=1 TO L
440 GOSUB 500:REM PROJECTIE
450 IF T(H)=0 THEN PLOT PX,PY
460 IF T(H)=1 THEN DRAWTO PX,PY
470 NEXT H
480 RETURN
490 REM PROJECTIE
500 PX=(X(H)*F+Y(H)*G+Z(H)*E)+125
510 PY=(X(H)*I+Y(H)*J+Z(H)*K)+75
520 RETURN
530 REM HOEKBEREKENING
540 A=(A1*3.14)/180
545 B=(B1*3.14)/180:C=(C1*3.14)/180
550 F=COS(B)*COS(C)
560 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
570 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
580 I=COS(B)*SIN(C)*-1
590 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
600 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
610 RETURN
```

De eerste regels zorgen voor een leeg scherm en vragen de gebruiker of hij zelf gegevens wil invoeren of gebruik wil maken van de al in het programma aanwezige kubus. Als er gebruik wordt gemaakt van de kubus springt de computer na het dimensioneren en inlezen van de arrays over naar regel 250 (regel 130).

De regels 140-240 zorgen ervoor dat de gebruiker zelf een vorm kan opgeven waarmee de computer vervolgens aan het werk kan gaan.

De regels 260-340 zorgen in een lus dat de gebruiker de draaihoeken op kan geven en zorgen door middel van twee subroutines voor het rekenen en voor het tekenen.

De regels 360-400 vormen de datalijst. Het gegeven achter de dataopdracht van regel 360 geeft alleen het aantal te lezen gegevens in de datalijst aan.

De regels 410-480 vormen de tekensubroutine. De grafische cursor wordt naar de juiste plaats gebracht (regel 420) en in een lus worden de gegevens een voor een afgewerkt. Eerst wordt het rekenwerk gedaan. Dit gebeurt in een aparte subroutine die op regel 490 begint. Aan deze rekenregels (500 en 510) kunt u zien dat er al wat rekenwerk is gedaan. Als er niet van te voren al wat uitgerekend was, had de computer voor elk punt moeten berekenen:

```
PX = X(H)*COS(B)*COS(C)+Y(H)*(SIN(A)*SIN(B)*COS(C)+COS(A)
     *SIN(C)) +Z(H)*(SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C))
PY = X(H)*-(COS(B)*SIN(C))+Y(H)*(COS(A)*COS(C)-SIN(A)*SIN(B)*
     SIN(C))+Z(H)*(COS(A)*SIN(B)*SIN(A)+SIN(A)*COS(C))
```

De berekeningen hiervoor staan grotendeels in de subroutine die buiten de tekenlus wordt aangeroepen, en de regels 530-610 beslaat.
Na de projectiesubroutine wordt gekeken of de laatste van de vier gegevens een 0 of een 1 is. Is het een 0 dan wordt er geen lijn getrokken, is het een 1, dan wordt er wel een lijn getrokken.

Een piramide

De mogelijkheden van het programmeren 3D zijn nog lang niet uitgeput. Neemt u het volgende programma over. Gebruik als basis het vorige programma, want er zijn veel overeenkomsten.

```
10 GRAPHICS 0
20 PRINT "DATALIJST (0) OF DATAINVOER (1)";
25 INPUT E
30 IF E=1 THEN 140
40 RESTORE
50 READ L
60 DIM X(L),Y(L),Z(L),T(L)
70 FOR H=1 TO L
80 READ A,B,C,D
90 X(H)=A:Y(H)=B
100 Z(H)=C:T(H)=D
120 NEXT H
130 GOTO 250
140 PRINT "AANTAL PUNTEN";:INPUT L
150 DIM X(L),Y(L),Z(L),T(L)
160 FOR H=1 TO L
170 ? "X-COORD. PUNT ";H;" ";:INPUT A
180 X(H)=A
190 ? "Y-COORD. PUNT ";H;" ";:INPUT B
200 Y(H)=B
210 ? "Z-COORD. PUNT ";H;" ";:INPUT C
220 Z(H)=C
230 PRINT "LIJN (1) OF VERPLAATS (0)";
235 INPUT D:T(H)=D
240 NEXT H
250 GRAPHICS 8+32
260 PRINT "   INVOER IN GRADEN"
265 PRINT "ROTATIE OM X-AS ";:INPUT A1
270 PRINT "ROTATIE OM Y-AS ";:INPUT B1
275 PRINT "ROTATIE OM Z-AS ";:INPUT C1
280 ? "SCHAAL X-RICHTING";:INPUT SX
285 ? "SCHAAL Y-RICHTING";:INPUT SY
290 ? "SCHAAL Z-RICHTING";:INPUT SZ
295 ? "VERPL. X-RICHTING";:INPUT VX
```

```
300 ? "VERPL. Y-RICHTING";:INPUT VY
305 GRAPHICS 8+16
310 GOSUB 540:REM HOEKBEREKENING
320 GOSUB 420:REM TEKENEN
330 FOR WA=1 TO 3000:NEXT WA
340 GOTO 250
350 END
360 DATA 12
370 DATA 0,0,0,0,50,0,0,1,50,0,50,1,0,0,50
375 DATA 1,0,0,0,1,25,-50,25,1,50,0,50,1
380 DATA 0,0,0,1,0,0,50,0,25,-50,25
385 DATA 1,50,0,0,1,0,0,50,1
410 REM TEKENEN
420 PLOT 125,125
430 FOR H=1 TO L
440 GOSUB 500:REM PROJECTIE
445 TRAP 1000
450 IF T(H)=0 THEN PLOT PX,PY
460 IF T(H)=1 THEN DRAWTO PX,PY
470 NEXT H
480 RETURN
490 REM PROJECTIE
500 PX=(X(H)*F*SX+Y(H)*G*SY+Z(H)*E*SZ)+125+VX
510 PY=(X(H)*I*SX+Y(H)*J*SY+Z(H)*K*SZ)+75+VY
520 RETURN
530 REM HOEKBEREKENING
540 A=(A1*3.14)/180
545 B=(B1*3.14)/180:C=(C1*3.14)/180
550 F=COS(B)*COS(C)
560 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
570 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
580 I=COS(B)*SIN(C)*-1
590 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
600 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
610 RETURN
1000 GRAPHICS 8+32
1010 ? "SORRY, DIT PAST NIET OP HET SCHERM."
1020 GOTO 250
```

Met de kennis van het voorgaande moet u dit programma helemaal kunnen volgen. De enige extra's zijn de mogelijkheden om een voorwerp te vervormen, te verkleinen of te vergroten. Verkleinen, vergroten of vervormen gebeurt allemaal op dezelfde manier. De opgegeven punten van het object worden gemanipuleerd door een vermenigvuldigingsfactor. Is deze 1 dan wordt het voorwerp in de opgegeven richting op ware grootte (de grootte uit de datalijst) getekend. Waarden groter dan 1 vergroten het voorwerp, waarden kleiner dan 1 verkleinen het.

Kiest u verschillende waarden voor de verschillende richtingen (X, Y en Z) dan vervormt u het voorwerp. De verplaatsing vindt plaats door de variabelen VX en VY.

De waarden die in de datalijst staan zorgen voor het tekenen van de omtrekken van een piramide. De waarden worden in groepen van 4 opgegeven. De eerste drie waarden geven een X-, Y- en Z-coordinaat. De vierde waarde geeft aan of er een lijn getrokken moet worden of dat naar een ander punt moet worden opgegeven. Op deze manier kunnen eenvoudig verschillende, ook ingewikkelde, voorwerpen getekend worden, zonder dat deze met een enkele ononderbroken lijn getekend hoeven te worden.
Het vinden van de benodigde gegevens is ook bij een niet symetrisch figuur niet moeilijk. U legt het voorwerp (of een maquette van het voorwerp) op een rasterpapiertje. De richting links-rechts noemt u de X-richting. Voor elk punt moet eerst de afstand van links naar rechts opgegeven worden.
De richting voor-achter noemt u de Z-richting.
Vlak voor het voorwerp (bij voorkeur op de X-as) plaatst u een stukje glas of een doorzichtige film/plastiek, met daarop een even groot rooster getekend. De richting boven-onder noemt u de Y-richting.
Op deze wijze kunt u van alle ter zake doende punten van het voorwerp de coordinaten bepalen. Bedenk bij het invoeren dat na de coordinaten van elk punt, telkens een 1 moet komen als er een lijn naar dat punt getrokken moet worden en een 0 als er alleen naar dat punt gesprongen moet worden. Heeft u heel ingewikkelde figuren, dan kunt u gebruik maken van een bochtenmeter. Dit voorwerp bestaat uit een stel ijzeren staafjes vlak naast elkaar. Door de verschuifbare staafjes allemaal tegen het voorwerp aan te leggen, de staafjes vast te zetten en de stand van de staafjes vervolgens op een roosterpapier af te lezen, zijn ook moeilijke vormen snel te bepalen. Dit stuk gereedschap is bij elke ijzerhandel verkrijgbaar.
Met behulp van dit programma en uw eigen fantasie kunt u veel voorwerpen die u goed kent, bekijken op het beeldscherm en onderzoeken of ze mooier zouden zijn als ze bijvoorbeeld wat langer, breder of hoger waren geweest.

BIJLAGE 1: DE ATASCII CODES

In de volgende tabel vindt u alle lettertekens die door de Atari gevormd kunnen worden. In de eerste kolom vindt u de decimale waarden. In de tweede kolom de tekens die de Atari kent. In de derde kolom vindt u de codes van de internationale ASCII lijst.

Decimale waarde	ATASCII teken	ASCII teken
0	♥	NUL
1	┣	SOH
2	┃	STX
3	┛	ETX
4	┫	EOT
5	┓	ENQ
6	╱	ACK
7	╲	BEL
8	◢	BS
9	▗	HT
10	◣	LF
11	▝	VT
12	▘	FF
13	▔	CR
14	▁	SO
15	▖	SI
16	♣	DLE
17	┏	DC1
18	━	DC2
19	╋	DC3
20	●	DC4
21	▄	NAK
22	▎	SYN
23	┳	ETB

Decimale waarde	ATASCII teken	ASCII teken
24	┻	CAN
25	▌	EM
26	┗	SUB
27	␛	ESC
28	↑	FS
29	↓	GS
30	←	RS
31	→	US
32		SPATIE
33	!	!
34	"	"
35	#	#
36	$	$
37	%	%
38	&	&
39	'	'
40	(	(
41	)	)
42	*	*
43	+	+
44	,	,
45	-	-
46	.	.
47	/	/

Decimale waarde	ATASCII teken	ASCII teken
48	0	0
49	1	1
50	2	2
51	3	3
52	4	4
53	5	5
54	6	6
55	7	7
56	8	8
57	9	9
58	:	:
59	;	;
60	<	<
61	=	=
62	>	>
63	?	?
64	@	@
65	A	A
66	B	B
67	C	C
68	D	D
69	E	E
70	F	F
71	G	G
72	H	H
73	I	I
74	J	J
75	K	K
76	L	L
77	M	M

Decimale waarde	ATASCII teken	ASCII teken
78	N	N
79	O	O
80	P	P
81	Q	Q
82	R	R
83	S	S
84	T	T
85	U	U
86	V	V
87	W	W
88	X	X
89	Y	Y
90	Z	Z
91	[	[
92	\	
93	]	]
94	^	^
95	_	_
96	●	
97	a	a
98	b	b
99	c	c
100	d	d
101	e	e
102	f	f
103	g	g
104	h	h
105	i	i
106	j	j
107	k	k

Decimale waarde	ATASCII teken	ASCII teken
108	l	l
109	m	m
110	n	n
111	o	o
112	p	p
113	q	q
114	r	r
115	s	s
116	t	t
117	u	u
118	v	v
119	w	w
120	x	x
121	y	y
122	z	z
123		
124	|	|
125		
126		
127		DEL
128		NUL
129		SOH
130		STX
131		ETX
132		EOT
133		ENQ
134		ACK
135		BEL
136		BS
137		HT

Decimale waarde	ATASCII teken	ASCII teken
138		LF
139		VT
140		FF
141		CR
142		SO
143		SI
144		DLE
145		DC1
146		DC2
147		DC3
148		DC4
149		NAK
150		SYN
151		ETB
152		CAN
153		EM
154		SUB
155	EOL	ESC
156		FS
157		GS
158		RS
159		US
160		SPATIE
161		!
162		"
163		#
164		$
165		%
166		&
167		'

Decimale waarde	ATASCII teken	ASCII teken
168	■	(
169	■	)
170	■	*
171	■	+
172	■	,
173	■	-
174	■	.
175	■	/
176	■	0
177	■	1
178	■	2
179	■	3
180	■	4
181	■	5
182	■	6
183	■	7
184	■	8
185	■	9
186	■	:
187	■	;
188	■	<
189	■	=
190	■	>
191	■	?
192	■	@
193	■	A
194	■	B
195	■	C
196	■	D
197	■	E

Decimale waarde	ATASCII teken	ASCII teken
198	■	F
199	■	G
200	■	H
201	■	I
202	■	J
203	■	K
204	■	L
205	■	M
206	■	N
207	■	O
208	■	P
209	■	Q
210	■	R
211	■	S
212	■	T
213	■	U
214	■	V
215	■	W
216	■	X
217	■	Y
218	■	Z
219	■	[
220	■	
221	■	]
222	■	
223	■	
224	■	–
225	■	a
226	■	b
227	■	c

Decimale waarde	ATASCII teken	ASCII teken
228	d	d
229	e	e
230	f	f
231	g	g
232	h	h
233	i	i
234	j	j
235	k	k
236	l	l
237	m	m
238	n	n
239	o	o
240	p	p
241	q	q

Decimale waarde	ATASCII teken	ASCII teken
242	r	r
243	s	s
244	t	t
245	u	u
246	v	v
247	w	w
248	x	x
249	y	y
250	z	z
251	♠	
252	|	|
253	↖	
254	◀	-
255	▶	DEL

BIJLAGE 2: FOUTMELDINGEN

Wanneer de Atari in een programma een fout ontdekt, stopt de computer met het programma en plaatst een foutmelding op het scherm. Deze foutmelding bestaat uit een code en (meestal) het regelnummer in het programma waar de fout werd ontdekt.
Het is mogelijk de computer een bepaalde taak te laten doen zodra er een fout wordt ontdekt. U gebruikt daarvoor de opdracht:

TRAP regelnummer

Achter TRAP plaatst u een regelnummer. Zodra de computer een fout in het programma tegenkomt, wordt er geen foutmelding gegeven, maar wordt er naar de opgegeven regel gesprongen. De programmeur moet daar opdrachten plaatsen, zodat de fout verbeterd wordt. Bijvoorbeeld: u heeft een programma waarbij u niet zeker weet of de grafische cursor niet in de loop van het programma aan de bovenkant van het scherm zal lopen (met andere woorden, de Y-coordinaat wordt kleiner dan 0). U kunt de foutmelding voorkomen door op een regel, voordat de fout mogelijkerwijs kan ontstaan, de opdracht TRAP 1000 te plaatsen. In regel 1000 plaatst u vervolgens de opdracht:

IF Y < 0 THEN Y=0 : RETURN.

Is de foutmelding veroorzaakt door een te kleine waarde voor Y, dan wordt de fout herstelt. Is er sprake van een andere fout, dan wordt de fout niet hersteld, en zal de computer het programma alsnog stop zetten en een foutcode op het scherm plaatsen.
De Atari kent de volgende foutcodes:

2 TE WEINIG GEHEUGENRUIMTE
Het programma, een opdracht, of het gebruik van variabelen vraagt teveel RAM geheugen. Dit kan vooral voorkomen bij grote arrays. Er zijn teveel FOR..NEXT lussen of subroutines genesteld.

3 VERKEERDE WAARDE
Een numerieke waarde is te groot, te klein of negatief terwijl een positieve waarde verwacht werd.

4 TE VEEL VARIABELEN
Een programma kan maximaal 128 verschillende variabelenamen bevatten. Komt u daarmee niet uit, probeer dan arrays te maken, of oude variabelenamen die niet meer gebruikt worden, opnieuw te gebruiken.

5 STRING TE LANG
Een deelstring geeft een letterteken op dat buiten de maximaal opgegeven (gedimensioneerde) stringlengte komt te liggen.

6 GEEN DATA GENOEG
Er zijn meer READ opdrachten dan er gegevens in de DATAlijst staan.

7 GETAL GROTER DAN 32767
Een opdracht die een geheel getal verwacht, krijgt als waarde een getal groter dan 32767 opgegeven.

8 VERKEERDE SOORT INPUT
Er is verschil tussen de gevraagde INPUT-soort en de opgegeven INPUT. Er wordt bijvoorbeeld om een numerieke waarde gevraagd en de gebruiker tikt een letter, een leesteken of een grafisch symbool in.

9 DIMENSIE FOUT
Een DIM opdracht bevat een array of string die al eerder geDIMensioneerd is. Er wordt een array of string gemaakt die groter is dan 32767 bytes. Het programma probeert een niet geDIMensioneerde array of string of een niet opgegeven array- of stringelement te gebruiken.

10 UITDRUKKING TE INGEWIKKELD
Een string- of numerieke berekening is te ingewikkeld en bevat teveel haakjes. Er zijn teveel genestelde GOSUB opdrachten.

11 DRIJVENDE KOMMA STROOMT OVER
Het programma probeert door 0 te delen. Op andere wijze wordt er een berekening uitgevoerd, die niet meer door de computer kan worden genoteerd. Het resultaat is groter dan 1E98 of kleiner dan 1E-99.

12 REGELNUMMER NIET TE VINDEN
Door een GOSUB, GOTO, IF..THEN, ON..GOSUB of ON..GOTO opdracht wordt verwezen naar een niet bestaande programmaregel.

13 NEXT ZONDER FOR
De computer komt in het programma een NEXT opdracht tegen, zonder eerst de bijhorende FOR opdracht te zijn tegen gekomen.

14 REGEL TE LANG
De opdracht is voor BASIC te lang of te ingewikkeld.

15 GOSUB- OF FOR-REGEL VERWIJDERD
Een RETURN of een NEXT opdracht kan de regel niet meer vinden, waar de bijhorende GOSUB of FOR stond. Deze regel is in de tussen liggende tijd verwijderd.

16 RETURN ZONDER GOSUB
Een RETURN opdracht werd tegen gekomen voordat de bijhorende GOSUB opdracht was uitgevoerd.

18 ONGELDIG STRING-LETTERTEKEN
Bij het uitvoeren van een VAL opdracht, had de om te zetten string geen numerieke waarden.

19 PROGRAMMA TE GROOT
Het programma dat wordt geladen, is te lang voor de RAM.

20 VERKEERD APPARAAT-NUMMER
Het programma probeert gebruik te maken van een niet toegestaan IOCB kanaal. Dat wil zeggen: kanaal 0 of een kanaal groter dan 7.

21 LOAD BESTANDFOUT
Het te laden programma of bestand is op een andere wijze dan met SAVE op cassette of diskette geplaatst (CSAVE of ENTER).

128 BREAK ONDERBREKING
U heeft op de BREAK toets gedrukt tijdens laden of opslaan van gegevens of een programma.

129 IOCB KANAAL IS AL OPEN
Het programma probeert een kanaal te openen dat al open is.

130 APPARAAT ONBEKEND
Het programma geeft een niet toegestane naam voor een apparaat op, of het apparaat is niet goed aangesloten.

131 ALLEEN UITVOER
Er wordt een GET of INPUT-opdracht gebruikt bij een kanaal dat alleen voor uitvoer geopend is.

132 XIO GRAMMATIKA FOUT
U heeft een XIO opdracht niet goed opgegeven.

133 KANAAL NIET OPEN
Het programma probeert een kanaal te gebruiken voordat het geopend is.

134 ONBEKEND KANAAL NUMMER
Het programma kan alleen de kanalen 1, 2, 3, 4, 5, 6 en 7 gebruiken.

135 ALLEEN INVOER
Een PRINT of PUT opdracht probeerden een kanaal te gebruiken dat alleen voor invoer was geopend.

136 EINDE VAN BESTAND
Het programma komt het punt tegen dat het einde van het bestand aangeeft, of probeert een deel van de diskette te lezen, dat niet bij het geopende kanaal hoort.

137 RECORD AFGEBROKEN
De computer komt een record tegen dat langer is dan 256 bytes, en breekt deze af.

138 APPARAAT ANTWOORD NIET
Ook na verschillende malen proberen, antwoordt een bepaald apparaat niet.

139 APPARAAT WERKT NIET
De programmarecorder of de diskdrive werkt niet goed of kan een bepaalde opdracht niet uitvoeren. Er wordt niet leesbare informatie over de seriele poort gestuurd.

140 FOUT BIJ SERIELE BUS
Er is een fout ontstaan bij het lezen van informatie via de seriele bus.

141 CURSOR BUITEN BEREIK
De positie van de cursor valt buiten de toegestane waarden bij de gebruikte schermmodus.

142 SERIELE FRAME FOUT
De gegevens van de seriele bus kloppen niet. De cassette of de diskette kan beschadigd zijn.

143 SERIELE CONTROLE KLOPT NIET
De gegevens waarmee de seriele bus worden gecontroleerd, kloppen niet. Er is een slechte opname op, of aflezing van de cassette of diskette. De cassette of diskette kan beschadigd zijn.

144 DISK FOUT
Het diskette systeem antwoordt niet naar behoren: De diskette is beschermd tegen wegschrijven, de inhoudsopgave van de diskette (directory) is fout of de diskdrive werkt niet goed.

145 LEES NA SCHRIJF FOUT
De diskdrive kwam een verschil tegen tussen wat er weggeschreven is en wat er weggeschreven moest worden. Verkeerde schermmodus.

146 NIET BESTAAND FUNKTIE
Het programma probeert via het toetsenbord uit te voeren, van de printer te lezen of een andere onmogelijke funktie uit te voeren.

147 NIET VOLDOENDE RAM VOOR GRAFIEK
Er is niet voldoende RAM geheugenruimte over voor de gewenste schermmodus.

150 SERIELE POORT OPEN
Elke seriele poort kan voor slechts een kanaal tegelijk geopend zijn.

151 SIMULTANE I/O MODUS
De seriele poort moet open zijn voordat XIO 40 mogelijk is.

152 ILLEGALE BUFFER
Bufferlengte en adres niet toegestaan.

153 SIMULTANE MODUS AL BEZIG
Het programma probeert invoer of uitvoer via een seriele poort te sturen terwijl er al een seriele poort open is voor concurrent modus.

154 SIMULTANE MODUS NIET AANWEZIG
De gevraagde invoer of uitvoer via een seriele poort vraagt om een simultane modus.

160 DRIVE NUMMER ONBEKEND
De enige mogelijk nummers voor disk-drives zijn: D:, D1:, D2:, D3: of D4:.

161 TEVEEL BESTANDEN OPEN
Normaal gesproken kunnen er maximaal drie disk-bestanden tegelijk open zijn.

162 DISK VOL
De diskette is vol.

163 FATALE I/O FOUT
Het invoer/uitvoer systeem is een fout tegen gekomen die niet meer overkomen kan worden. U moet de computer uit en weer aan zetten.

164 BESTANDSNUMMER KLOPT NIET
De informatie op de diskette die door de computer wordt gebruikt. voor het uitwisselen van gegevens met de diskette is niet meer intakt.

165 SLECHTE BESTANDSNAAM
Het programma geeft een bestandsnaam op die niet toegestaan is.

166 POINT LENGTE FOUT
De POINT opdracht wijst naar een niet bestaande byte in een sector.

167 BESTAND OP SLOT
Het programma probeert een bestand dat gesloten is te veranderen of te wissen.

168 ONGELDIGE XIO OPDRACHT
De XIO opdracht is ongeldig, of niet juist samengesteld.

169 INHOUDSOPGAVE (DIRECTORY) VOL
De inhoudsopgave van een diskette kan maximaal 64 verschillende namen bevatten. Dit is onafhankelijk van de hoeveelheid ruimte die de bestanden en

programma's op de diskette innemen. Het is mogelijk dat de schijf eerder vol is dan de inhoudsopgave, of andersom.

170 BESTAND NIET GEVONDEN

De opgegeven bestandsnaam staat niet in de inhoudsopgave van de diskette die nu in de gebruikte disk-drive zit.

171 POINT ONGELDIG

Het programma probeerde met een POINT opdracht naar een disk-sector te wijzen die niet bij het geopende bestand hoort.

BIJLAGE 3: BASIC STATEMENTS

Statements is de verzamelnaam voor opdrachten en funkties. In het algemeen geeft een funktie een resultaat (een string of een getal) en vervult een opdracht een bepaalde taak.
Deze lijst met BASIC statements behandelt alleen de statements die voorkomen in standaard Atari-BASIC.
Voor de meeste statements geldt dat u voor een meer uitgebreide uitleg verwezen wordt naar elders in dit boek. Zie ook de index.
Na de alfabetische lijst van statements, vindt u 6 tabellen met de statements gegroepeerd naar onderwerp. In deze tabellen worden tevens de minimale afkortingen van de statements gegeven.
Overal waar bij de behandeling van een statement het gebruik van een variabele nodig was, is voor de variabele x of y gekozen. Dit verwijst niet naar de grafische coordinaten.

ABS(x)

Een numerieke funktie die als resultaat de absolute waarde van x geeft. De opgegeven waarde moet een getal, een variabele of een expressie zijn.
Bijvoorbeeld:

```
PRINT ABS(-3),ABS(7-10),ABS(10-7)
```

Resultaat: 3, 3, 3.

ADR(x$)

Een funktie om het geheugenadres te vinden van het eerste letterteken van de string x$. Bij het dimensioneren van een string, reserveert de Atari een vast gebied in het geheugen voor de string. Door de ADR opdracht kan bekeken worden op welk adres de string precies begint.

Voorbeeld: `PRINT ADR(Q$)`

AND

Een logische operator, gebruikt in voorwaardelijke statements. Bijvoorbeeld:

```
IF uitdrukking1 AND uitdrukking2 THEN statement
```

De statement achter THEN wordt alleen uitgevoerd als zowel uitdrukking1 als uitdrukking2 ware beweringen zijn.

Voorbeeld: `IF X > 0 AND X < 0 THEN PRINT "onzin"`

ASC(x$)

Een stringfunktie die de ASCII-code geeft van het eerste letterteken in de opgegeven string (x$). Als de x$ leeg is (een zogenaamde nul-string) geeft de funktie een ASCII waarde van 44 als resultaat.

Voorbeeld: IF ASC(INVOER$) < 78 THEN GOTO 1000

ATN(x)

Een numerieke funktie die als resultaat de arctangens van x geeft. x moet een getal of een expressie (berekening) zijn.
De arctangens is het omgekeerde van een tangens. Dat wil zeggen dat de arctangens van een getal gelijk is aan de hoek waarvan de tangens gelijk is aan dat getal.
De Atari werkt met radialen. Door de opdracht DEG schakelt de Atari over op het werken met graden.

Voorbeeld: PRINT ATN(HOEK)

BREAK

Dit wordt gebruikt als toetsindruk om doorgang van programma te stoppen. Zowel de geluidsgeneratoren als de invoer/uitvoerkanalen blijven normaal in werking. Door CONT in te tikken, hervat het programma zijn werking. Dit werkt alleen als er geen enkele andere opdracht tussen het indrukken van BREAK en het intikken van CONT is gegeven.

BYE

Een opdracht die van Atari BASIC overschakelt naar testmodus (u kunt kiezen uit drie testen). Het aanwezige BASIC programma wordt niet beinvloed. Evenmin de variabelen, DOS of RS-232. U keert weer terug naar BASIC door het indrukken van de SYSTEM RESET toets.

Voorbeeld: BYE

CLOAD

Behandelt de invoer van een eerder opgenomen programma van de cassetterecorder in het geheugen.
CLOAD opent eerst kanaal 7 voor invoer vanaf de programmarecorder. Als kanaal 7 al open is naar een ander apparaat, krijgt u een foutmelding. Deze foutmelding zorgt ervoor dat het kanaal gesloten wordt, zodat een tweede poging met CLOAD wel succesvol zal zijn.
Bij het uitvoeren van de CLOAD opdracht laat de computer een enkele piep horen. U dient nu de cassette door middel van de FAST FORWARD en REWIND op de juiste positie te plaatsen. Zodra u de juiste plaats heeft

gevonden, drukt u de PLAY knop van de programmarecorder in en drukt u op een willekeurige toets (behalve de BREAK) van het toetsenbord.
U hoort nu even niets, en daarna enkele signalen van de luidspreker: de computer is het programma aan het laden. Als het stil wordt, is het programma geladen (de READY melding verschijnt op het scherm).
CLOAD werkt alleen met programma's die door CSAVE of SAVE op een cassette zijn opgenomen. Een programma dat door middel van LIST opgeslagen is, zal niet door CLOAD geladen kunnen worden.
CLOAD houdt automatisch de NEW opdracht in. Zelfs voordat de luidspreker klinkt, zijn alle programmaregels en variabelen uit het geheugen gewist. Het onderbreken van CLOAD door middel van BREAK stopt wel de CLOAD opdracht, maar zorgt er niet voor dat het uit het geheugen gewiste programma terugkeert.

Voorbeeld: CLOAD

CHR$(x)

Geeft de stringwaarde van de opgegeven ASCII code. x moet een getal of een uitdrukking met als resultaat een getal tussen 0 en 255 zijn. Een getal kleiner dan 65536 zal niet tot een foutmelding leiden: de computer trekt er een veelvoud van 256 af.

Voorbeeld: FOR A=33 TO 100 : PRINT CHR$(A) : NEXT A

CLOG(x)

Een numerieke funktie die de logaritme van x geeft. x is een getal of een numerieke expressie met een resultaat groter dan 0.
Een 0 of een getal of expressie kleiner dan 0 geeft een foutmelding.

Voorbeeld: X=A*CLOG(Q)

CLOSE #x

Sluit kanaal x. x moet een bestaand kanaalnummer zijn (0-7).
De END opdracht sluit alle IOCB kanalen, behalve 0 omdat deze normaal gesproken met het scherm en toetsenbord is verbonden.

Voorbeelden: CLOSE #1
CLOSE #Z

CLR

Kent de waarde 0 toe aan alle variabelen. Van arrays en strings worden de dimensies nul. De DATA lijstwijzer wordt ingesteld op nul.
De tabel met alle in het geheugen voorkomende variabelenamen wordt niet gewist. Daarvoor zorgt alleen de NEW opdracht. De CLR opdracht helpt niet

als u een foutmelding krijgt doordat u meer dan 128 verschillende variabelenamen probeert te gebruiken.
Na de CLR opdracht moeten arrays en strings opnieuw worden gedimensioneerd. worden.

Voorbeeld: CLR

COLOR x

Een grafische funktie die aangeeft welk kleurregister gebruikt gaat worden bij de komende PLOT en DRAWTO opdrachten. x moet een positieve numerieke waarde of expressie zijn.
De relatie tussen de waarde die in de COLOR opdracht gegeven wordt en de uiteindelijke op het scherm komende kleur is afhankelijk van de gebruikte schermmodus.
De COLOR opdracht verwijst niet altijd naar dezelfde registers:

SETCOLOR kleurregister	COLOR opdracht bij scherm: 3, 5, 7, 15	4, 6, 14	8
0	1	1	-
1	2	-	1
2	3	-	0
3	-	-	-
4	0	0	-

De standaard instellingen van de verschillende kleurregisters is als volgt:

Kleurregister	kleur	helderheid	kleur
0	2	8	oranje
1	12	10	groen
2	9	4	donker blauw
3	4	6	rood
4	0	0	zwart

In de GTIA schermen 9, 10 en 11 werken de COLOR opdrachten anders. In scherm 10 verwijst de COLOR opdracht naar een bepaalde geheugenplaats, waar een kleur + helderheid geplaatst kan zijn:

POKE 704, 16*kleur+helderheid bepaalt	COLOR 0 (achtergrond)
POKE 705, ,, ,,	COLOR 1
POKE 706, ,, ,,	COLOR 2
POKE 707, ,, ,,	COLOR 3
POKE 708, ,, ,,	COLOR 4
POKE 709, ,, ,,	COLOR 5
POKE 710, ,, ,,	COLOR 6
POKE 711, ,, ,,	COLOR 7
POKE 712, ,, ,,	COLOR 8

In scherm 11 verwijst de COLOR opdracht naar een van de standaard kleuren van de Atari. De opdracht SETCOOR 4,0, helderheid bepaalt de helderheid van alle gebruikte kleuren:

COLOR (in scherm 11)	Kleur
0	grijs
1	licht oranje
2	oranje
3	rood-oranje
4	roze
5	paars
6	paars-blauw
7	blauw
8	blauw
9	licht blauw
10	blauw-groen
11	groen-blauw
12	groen
13	geel-groen
14	oranje-groen
15	licht oranje

In het tekstscherm 0 kan de COLOR-opdracht worden gebruikt om te bepalen welk letterteken door de PLOT of DRAWTO opdracht getekend gaat worden. De PLOT en DRAWTO opdrachten kunnen in schermmodus 0 gebruikt worden om op willekeurige plaatsen een enkel teken of een serie tekens te plaatsen. Onderstaande tabel geeft de waarden die de COLOR opdracht moet hebben om de tekens weer te geven bij een PLOT of DRAWTO opdracht. Het eerste getal zorgt voor normale weergave, het tweede getal voor inverse weergave.

COLOR in schermmodus 0											
Waarde nor/inv	tek.	waarde nor/inv	tek.	waarde nor/inv	tek.	waarde nor/inv	tek.	waarde nor/inv	tek.	waarde nor/inv	tek.
0 128	♥	23 151		46 174	.	69 197	E	92 220	\	115/243	s
1 129		24 152		47 175	/	70 198	F	93 221	]	116 244	t
2 130		25 153		48 176	0	71 199	G	94 222	^	117 245	u
3 131		26 154		49 177	1	72 200	H	95 223	_	118 246	v
4 132		27 —		50 178	2	73 201	I	96 224	♦	119 247	w
5 133		28 156	↑	51 179	3	74 202	J	97 225	a	120 248	x
6 134		29 157	↓	52 180	4	75 203	K	98 226	b	121 249	y
7 135		30 158	←	53 181	5	76 204	L	99 227	c	122 250	z
8 136		31 159	→	54 182	6	77 205	M	100 228	d	123 251	♠
9 137		32 160		55 183	7	78 206	N	101 229	e	124 252	|
10 138		33 161	!	56 184	8	79 207	O	102 230	f	125/—	clr scrn
11 139		34 162	"	57 185	9	80 208	P	103 231	g	126 254	◀
12 140		35 163	#	58 186	:	81 209	Q	104 232	h	127/255	▶
13 141		36 164	$	59 187	;	82 210	R	105 233	i	— 155	EOL
14 142		37 165	%	60 188	<	83 211	S	106 234	j	— / 253	
15 143		38 166	&	61 189	=	84 212	T	107 235	k		
16 144		39 167	'	62 190	>	85 213	U	108 236	l		
17 145		40 168	(	63 191	?	86 214	V	109 237	m		
18 146		41 169	)	64 192	@	87 215	W	110 238	n		
19 147		42 170	*	65 193	A	88 216	X	111 239	o		
20 148		43 171	+	66 194	B	89 217	Y	112 240	p		
21 149		44 172	,	67 195	C	90 218	Z	113 241	q		
22 150		45 173	-	68 196	D	91 219	[	114 242	r		

In de tekstmodi 1, 2, 12 en 13 kan de COLOR opdracht worden gebruikt om een bepaald teken in een van vier gewenste kleuren op het scherm te zetten met PLOT of DRAWTO. In de tabel hieronder staan voor elk letterteken vier waarden. Elk van deze waarden achter de COLOR opdracht zorgt voor het gewenste letterteken. De eerste waarde geeft het teken weer in de kleur van kleurregister 0, het tweede getal geeft het teken weer in de kleur van kleurregister 1, het derde getal geeft het teken weer in de kleur van kleurregister 2 en het laatste getal geeft het teken weer in de kleur die opgeslagen is in kleurregister 3. Voor het veranderen van de kleurwaarden in de kleurregisters, zie SETCOLOR.

De genoemde tekstmodi kennen twee series met lettertekens. De linkse tekens

komen normaal gesproken op het scherm. Om de tekens in de rechter rijen te krijgen dient u eerst als opdracht te geven:

POKE 756,226

De standaard lettertekens krijgt u weer met:

POKE 756,224

Waarde voor kleurregister				Teken		Waarde voor kleurregister				Teken	
0	1	2	3			0	1	2	3		
32	0	160	128	(spatie)	♥	64	96	192	224	@	◆
33	1	161	129	!	├	65	97	193	225	A	a
34	2	162	130	"	▕	66	98	194	226	B	b
35	3	163	131	#	┘	67	99	195	227	C	c
36	4	164	132	$	┤	68	100	196	228	D	d
37	5	165	133	%	┐	69	101	197	229	E	e
38	6	166	134	&	╱	70	102	198	230	F	f
39	7	167	135	'	╲	71	103	199	231	G	g
40	8	168	136	(	◢	72	104	200	232	H	h
41	9	169	137	)	▗	73	105	201	233	I	i
42	10	170	138	*	◣	74	106	202	234	J	j
43	11	171	139	+	▝	75	107	203	235	K	k
44	12	172	140	,	▘	76	108	204	236	L	l
45	13	173	141	-	▔	77	109	205	237	M	m
46	14	174	142	.	▁	78	110	206	238	N	n
47	15	175	143	/	▖	79	111	207	239	O	o
48	16	176	144	0	♣	80	112	208	240	P	p
49	17	177	145	1	┌	81	113	209	241	Q	q
50	18	178	146	2	─	82	114	210	242	R	r
51	19	179	147	3	┼	83	115	211	243	S	s
52	20	180	148	4	●	84	116	212	244	T	t
53	21	181	149	5	▄	85	117	213	245	U	u
54	22	182	150	6	▎	86	118	214	246	V	v
55	23	183	151	7	┬	87	119	215	247	W	w
56	24	184	152	8	┴	88	120	216	248	X	x
57	25	185	153	9	▌	89	121	217	249	Y	y
58	26	186	154	:	└	90	122	218	250	Z	z
59	27	187		;	␛	91	123	219	251	[	♠
60	28	188	156	<	↑	92	124	220	252	\	|
61	29	189	157	=	↓	93		221	253	]	↰
62	30	190	158	>	←	94	126	222	254	^	◀
63	31	191	159	?	→	95	127	223	255	_	▶

Voorbeelden: COLOR 1
COLOR Q
COLOR Z+2*Q

COM x$(x),y$(y)
COM x1(y1,z1),x2(y2,z2)

Deze opdracht is gelijk aan de DIM opdracht. Reservering van voldoende geheugenruimte voor strings en arrays. Zie ook bij DIM.

Voorbeelden: COM X(12),Y(13),Z$(5)

CONT
Een opdracht om het programma na bepaalde onderbrekingen weer voort te zetten.
Na een onderbreking door BREAK, END of STOP gaat het programma door een CONT weer verder.
In samenhang met de BREAK toets kan de listing van een programma gestopt en weer voortgezet worden (dit is ook mogelijk door telkens de CONTROL en de 1 gelijk in te drukken).

Het programma wordt door een CONT opdracht voort gezet bij de eerste statement van de programmaregel volgend op de programmaregel waarin de stopzettende opdracht voorkwam. Indien er op de regel met de STOP opdracht nog opdrachten achter de STOP opdracht stonden, worden deze over geslagen.

Voorbeeld: CONT

COS(x)

Een numerieke funktie die de cosinus van een hoek, aangegeven door x, als resulaat heeft. x moet een getal of een numerieke uitdrukking zijn. x wordt normaal gesproken in radialen uitgedrukt. Na een DEG opdracht wordt x gelezen als in graden uitgedrukt.

Voorbeeld: PLOT X,COS(Z)*50

CSAVE
Met deze opdracht schrijft u het in het geheugen aanwezige BASIC programma in gecodeerde vorm weg op een cassette.
De computer begint bij het uitvoeren van de opdracht met het afgeven van twee korte piepjes. Plaats nu de cassette op de juiste positie in de cassetterecorder. U kunt de FAST FORWARD en REWIND knop vrijelijk gebruiken. Druk als de cassette op de juiste plaats staat, de PLAY en RECORD (=opname) knop in en druk op een willekeurige toets van het toetsenbord (niet de BREAK toets). De computer schakelt nu alle geluidsregisters die nog aanstaan uit, en stuurt het programma naar de cassette. Dit is hoorbaar als een

verzameling verschillende geluiden. Zodra het programma over is gestuurd, wordt het stil. De READY mededeling verschijnt op het scherm.
CSAVE gebruikt kanaal 7 voor de communicatie met de programmarecorder. Is dit kanaal al open, dan krijgt u een foutmelding. De foutmelding zorgt er voor dat het kanaal gesloten wordt. Een tweede poging zal wel succesvol zijn.
De CSAVE opdracht slaat het programma in gecodeerde vorm op. Alleen een CLOAD opdracht kan het programma weer van de cassette lezen. De ENTER of LOAD opdrachten werken niet met een gecodeerd programma.
Als u de CSAVE opdracht onderbreekt door een BREAK, is het programma op de cassette incompleet. Zo'n incompleet programma kan niet van cassette worden gelezen.

Voorbeeld: CSAVE

DATA x,y,z,..

De DATA opdracht vormt een lijst met gegevens die door READ opdrachten aan variabelen kunnen worden toegekend.
Achter de DATA opdracht worden strings, numerieke waarden of beide geplaatst. Deze waarden worden gelezen door een of meer READ opdrachten. De string gegevens hoeven niet omgeven te zijn door aanhalingstekens.
De string en numerieke gegevens mogen door elkaar in de lijst staan, maar het is belangrijk dat de READ opdracht het juiste soort gegeven leest. De gegevens worden een voor een uit de lijst gelezen. Daarna wordt de datawijzer verplaatst naar het volgende gegeven, dat dan door de volgende READ opdracht gelezen zal worden.
De gegevens worden gescheiden door komma's.
Omdat komma's de verschillende gegevens scheiden, mogen deze niet voorkomen in stringgegevens. Verder mogen strings alle lettertekens, leestekens en getallen bevatten. Twee komma's onmiddellijk na elkaar worden door de computer opgevat als een string met een lege inhoud.
De DATA-opdrachten mogen op elke plaats in het programma worden geplaatst.
Na een DATA opdracht mogen op dezelfde regel geen andere BASIC statements meer volgen.
U kunt een DATA opdracht als direkte opdracht geven. Er zal geen foutmelding volgen. De opgegeven data kunnen dan niet door een READ opdracht worden gelezen. Het is een zinloze opdracht.

Voorbeelden:

```
DATA voorbeeld, voorbeeld2, test, string
DATA 10, 30, 50, 60,20
DATA jan, 10, piet, 60, klaas, 20
DATA 6+8
```

DEG

Deze opdracht vertelt de computer vanaf dat moment met graden te werken in plaats van radialen. Om terug te keren naar het werken met radialen gebruikt u

de opdracht RAD, drukt u op SYSTEM RESET, zet u de machine uit en aan, tikt u RUN of tikt u NEW.

Voorbeeld: DEG

DIM x$(x),y$(y),...
DIM x1(y1,z1),x2(x2(y2,z2),...

De DIM-opdracht reserveert voldoende geheugenruimte in de computer voor een string of voor een array. Voordat in een programma gebruik wordt gemaakt van een string of een array, moet deze eerst gedimensioneerd zijn. Atari BASIC staat arrays in maximaal twee dimensies toe.
Als in het programma naar een array-element wordt verwezen, mag de index van de variabel (de plaats in de array) niet kleiner zijn dan 0 en niet groter dan de lengte van de array.
Een string moet tot zijn maximale lengte gedimensioneerd worden. Het is niet mogelijk de lengte van een string later in het programma te verlengen (behalve na een CLR opdracht, zie daar). Een poging een string langer te maken dan de gedimnensioneerde lengte zal niet lukken.
De maximale lengte voor een enkele string is 32767 lettertekens. Daarbij wordt de totale lengte die wordt gebruikt voor het dimensioneren van strings en arrays beperkt door de geheugenruimte van de computer.

Voorbeeld: DIM A$(12),B$(78),ARRAY(12,15)
DIM NAAM$(15),ADRES1$(15),ADRES2$(15),
ADRES3$(15)

DOS

Laadt het DOS programma in het geheugen en geeft het DOS menu weer op het scherm. Om naar BASIC te gaan kiest u uit het menu mogelijk B of drukt u op de SYSTEM RESET knop.
Indien u een disk-drive heeft, moet u altijd voordat u met programmeren begint, het DOS programma laden.
Zie de bijlage DISK-DRIVE

Voorbeeld: DOS

DRAWTO x,y

Een grafische opdracht die een lijn trekt van de huidige positie van de grafische cursor naar het opgegeven punt (x,y).
De meest recente COLOR-opdracht bepaalt in welke kleur de lijn wordt getekend. Als er geen COLOR-opdracht is geweest, wordt de lijn in de kleur van de achtergrond getekend (en is dus onzichtbaar).
In de tekstschermmodi wordt geen lijn getrokken, maar een rij lettertekens die begint bij de laatste op het scherm gezette letter en eindigend bij het opgegeven punt.

Voorbeelden: DRAWTO 10,15
DRAWTO KOLOM,RIJ

END

Geeft het einde van een BASIC programma aan. Als de laatste uit te voeren opdracht het hoogste regelnummer heeft, is de opdracht niet nodig.
Alle IOCB kanalen, behalve 0, worden gesloten. De geluidsgeneratoren worden uit gezet.

Voorbeeld: END

ENTER apparaat

Mengt een ATASCII programma vanaf het apparaat in het geheugen van de computer. Het programma dat zich al in het geheugen bevindt, wordt niet uitgewist. De nieuwe programmaregels worden bij de oude programmaregels geplaatst. Zijn er programmaregels van het oude en nieuwe programma die hetzelfde nummer hebben, dan schrijven de nieuwe nummers over de oude heen. De oude nummers vervallen daarmee.
De tabel met variabelenamen wordt niet gewist. Nieuwe variabelenamen van het nieuw te mengen programma worden extra in de tabel geplaatst.
Alleen programma's die door middel van LIST zijn opgeslagen op cassette of diskette, kunnen met ENTER weer worden ingevoerd. ENTER werkt niet met programma's die door SAVE of CSAVE in een gecodeerde vorm op de cassette of diskette zijn geschreven.

Voorbeelden: ENTER "C:"
ENTER "D:PROGRAMMA.BAS"

EXP(x)

Een numerieke funktie die de constante e (ongeveer 2.71828182) tot de macht x als resultaat geeft. x moet een getal of een numerieke uitdrukking zijn.

Voorbeeld: Q=EXP(Z)

FOR x = begin TO einde (STEP y)

Deze statement definieert een FOR..NEXT lus. Een numerieke variabele (x) wordt van 'begin' tot 'einde' verhoogd met 1, of indien de STEP statement is gebruikt, met de STEPgrootte (y).
Bij deze opdracht hoort altijd een NEXT opdracht.

De waarde van x wordt telkens na het verhogen (met 1 of met de STEP grootte) vergeleken met de 'einde' waarde achter FOR. Zodra x groter is dan 'einde', wordt uit de lus gesprongen en vervolgt de computer het programma bij de regel volgend op de regel met de NEXT opdracht. Omdat het vergelijken

plaats vindt na de verhoging van x, wordt de lus altijd minimaal een keer doorlopen.
FOR..NEXT lussen kunnen worden genesteld ,dat wil zeggen dat er verschillende lussen binnen elkaar komen te liggen. De lussen mogen elkaar niet kruisen. Elke lus moet een unieke variabelenaam hebben. Omdat Atari BASIC het gebruik van maximaal 128 verschillende variabelenamen toelaat, kunt u maximaal 128 lussen binnen elkaar maken. Dit kost erg veel geheugenruimte. De waarden voor het begin van de lus, het einde van de lus en de STEP grootte, worden alleen de eerste keer dat de computer de FOR opdracht tegen komt, genoteerd door de computer. Als u deze waarden binnen de lus verandert, heeft dat geen effect voor de werking van de lus.
U kunt de lus voortijdig verlaten door de variabele een waarde te geven die gelijk is aan de 'einde' waarde.
Bijvoorbeeld:

```
FOR Q=1 TO 10
READ A
IF A=999 THEN Q=10
NEXT Q
```

Deze lus onderzoekt de waarden in een DATA lijst. Zodra de computer een waarde van 999 tegenkomt, wordt de waarde van de lus-variabele gelijk gemaakt aan de maximale waarde in de lus. Als de waarde van Q nu wordt vergeleken met de maximale waarde voor de lus, ziet de computer dat deze is bereikt, en wordt er uit de lus gesprongen.

U kunt een FOR..NEXT lus gebruiken in de direkte modus. De gehele lus moet dan binnen een enkele ingetikte regel vallen. Vergeet u de NEXT opdracht, dan wordt de lus een keer uitgevoerd.

Voorbeeld:

```
FOR TELLER= 1 TO 100
FOR ACHTERUIT= 100 TO 1 STEP -1
FOR A = B TO C STEP D
```

Let op de volgende punten bij het gebruik van FOR..NEXT lussen:
- Begin geen lus buiten een subroutine die binnen een subroutine eindigt.
- Maak geen vertakkingen die binnen een lus eindigen. Een lus moet beginnen met een FOR opdracht.
- Maak geen vertakking die zomaar uit een lus springt. Er blijft dan een ongebruikte NEXT opdracht over. Dit kost zeer veel geheugenruimte.

U kun het overblijven van een ongebruikte NEXT opdracht voorkomen door de POP opdracht. Deze opdracht zorgt ervoor dat de computer 'vergeet' dat hij bezig was met een FOR..NEXT lus uit te voeren. Zie onder POP.

FRE(x)

Een numerieke funktie die het aantal nog vrij beschikbare bytes aan RAM geeft. De variabele x is een 'dummy' variabele. Het doet er niet toe welke

waarde u hier invult, zolang het maar een getal of een numerieke uitdrukking is.
Bijvoorbeeld:

```
PRINT FRE(0)
```

resultaat: het aantal bytes in RAM die u nog voor uw programma kunt gebruiken.

GET #x,variabele

GET neemt een enkele byte aan informatie van het IOCB kanaal nummer x. De informatie wordt toegekend aan de variabele achter de komma. De informatie (string of numeriek) moet overeenstemmen met het soort variabele achter de komma.

Als u de GET opdracht gebruikt om informatie van het toetsenbord te krijgen, krijgt u als gegevens de decimale ATASCII waarden van de toetsen.
Bijvoorbeeld:

```
10 OPEN #1,4,0,"K:"
20 GET #1,TOETS
30 PRINT CHR$(TOETS);" = ";TOETS
40 GOTO 20
```

Dit programma geeft u telkens de ATASCII waarde van een toets, samen met de letter die bij die toets hoort.

U kunt de GET opdracht ook gebruiken om informatie van de cassetterecorder te lezen. De computer leest informatie van de cassetterecorder in blokken van 128 bytes. Zodra u een kanaal opent om gegevens vanaf de cassetterecorder te lezen, zorgt de eerste GET opdracht ervoor dat er 128 bytes van de cassette in de cassettebuffer van de computer worden geladen. De eerste byte van deze groep van 128 wordt toegekend aan de variabele achter de komma van de GET opdracht, en de cassetterecorder wordt stilgezet. De eerste waarde wordt uit de buffer gehaald. Zodra alle 128 bytes uit de buffer zijn gehaald, wordt de cassetterecorder weer aangezet om de volgende 128 bytes te lezen. Als u meer bytes probeert te lezen dan er in het bestand aanwezig zijn, leidt dit tot een foutmelding. Het invoerkanaal kan worden gesloten door een CLOSE of een END opdracht.
Het gebruik van de GET opdracht met een disk-drive werkt vergelijkbaar met het gebruik bij een cassetterecorder. Nu worden er echter geen 128 bytes tegelijk gelezen, maar net zoveel bytes als er in een sector op de diskette passen. Bij de Atari 810 disk-drive is dat 125 bytes.
De GET opdracht kan bytes lezen die op de diskette geplaatst zijn door een PUT opdracht of door een PRINT opdracht. In dat laatste geval wordt de informatie door de PRINT-opdracht met meer dan een byte tegelijk op de diskette geplaatst. De GET-opdracht leest echter slechts een byte tegelijk van

de diskette. De waarden die door de GET opdracht gelezen worden zijn de ATASCII-codes van de tekens die door de PRINT opdracht zijn geplaatst. Met de POINT-opdracht kunt u de computer van de ene sector naar de andere sector laten springen.

De GET-opdracht kan tevens gebruikt worden om de code te lezen van het teken of de grafische punt op de huidige positie van de tekstcursor of grafische cursor. (S: of E:). De verkregen code is gelijk aan de codes die gebruikt worden door de COLOR opdracht. Zie de tabellen daar.
Nadat de GET-opdracht de code gevonden heeft, wordt de cursorpositie aangepast, zodat het volgende teken of het volgende grafische punt bekeken kan worden. Als u een GET opdracht probeert uit te voeren nadat het laatste punt van de laatste regel is bekeken, volgt een foutmelding.

Voorbeelden: GET#1,X
GET#KANAAL, Z

GOSUB regelnummer

Dit is een programma-besturings statement. De uitvoering van het programma wordt naar het opgegeven regelnummer gestuurd. Zodra de computer een RETURN opdracht tegenkomt, wordt het programma hervat bij de regel volgend op de regel met de GOSUB-opdracht.
De opgegeven regel moet in het programma bestaan, anders volgt een foutmelding.
Ook voor een subroutine geldt dat het niet raadzaam is de subroutine te verlaten zonder de RETURN opdracht uit te voeren. Een niet gebruikte RETURN opdracht kost veel geheugenruimte. U kunt dit opheffen door een POP opdracht; de computer vergeet dan dat hij bezig was met het uitvoeren van een subroutine.
Een GOSUB opdracht mag overal in het programma voorkomen. Een subroutine moet aan het begin van een regel beginnen.

Voorbeelden: GOSUB 1000
GOSUB X
IF A=7 THEN GOSUB A*10

GOTO regelnummer

Dit is een programma-besturings statement. De uitvoering van het programma wordt onvoorwaardelijk en zonder terugkeer naar een bepaalde programmaregel gestuurd. 'regelnummer' moet een positief geheel getal of een expressie met als resultaat een positief geheel getal, zijn. De opgegeven regel moet in het programma voorkomen.

Voorbeelden: GOTO 1250
GOTO Z

GRAPHICS x

Deze opdracht stelt een bepaalde schermmodus in. x moet een getal of een uitdrukking zijn die een geldig nummer voor een schermmodus oplevert. Als x tussen 0 en 15 ligt, krijgt u een schermmodus met een tekstvenster (niet bij alle schermmodi).
Door 16 op te tellen bij x krijgt u dezelfde schermmodus, maar zonder tekstvenster.
Door 32 op te tellen bij x krijgt u dezelfde schermmodus, met tekstvenster, maar het scherm wordt niet gewist.
Door 48 op te tellen bij x krijgt u dezelfde schermmodus, zonder tekstvenster en het scherm wordt niet gewist.

Door de GRAPHICS opdracht wordt het scherm ingesteld, wordt er voldoende geheugenruimte vrij gemaakt voor het gevraagde scherm, wordt de tekstcursor op de (0,0) positie geplaatst, wordt de grafische cursor op (0,0) geplaatst en worden de kleurregisters op hun standaardwaarden ingesteld.

Zie voor een uitgebreide behandeling van de verschillende schermmodi het hoofdstuk 'Schermen'.

Voorbeelden:
```
GRAPHICS 8
GRAPHICS 8+16
GRAPHICS A+32
```

IF uitdrukking THEN statement

De IF..THEN opdracht is de belangrijkste voorwaardelijke statement van BASIC. Voluit betekent de statement: IF de uitdrukking is waar THEN voer de statement uit. Dat wil zeggen dat als de uitdrukking niet waar is, de statement(s) achter THEN niet word(en) uitgevoerd en de computer het programma vervolgt met de volgende programmaregel.
Zie hoofdstuk 8: Logica en IF..THEN.

Voorbeelden:
```
IF A$ <> "J" THEN GOTO 1200
IF Q < 12 THEN PRINT "KLEINER"
IF A=B THEN C=D
IF Q THEN PRINT "Q is waar"
```

INPUT x,y
INPUT x$,y$
INPUT #kanaal, x,y

De INPUT opdracht zorgt voor invoer van gegevens. De normale INPUT opdracht drukt een vraagteken af op het scherm en wacht tot de gebruiker informatie heeft ingetikt, en deze informatie heeft afgesloten door het indrukken van de RETURN toets.

De INPUT opdracht kan een waarde direkt aan een variabele toekennen. Bijvoorbeeld:

```
INPUT X
```

of

```
INPUT X$
```

In het bovenste voorbeeld zal de ingetikte waarde aan de numerieke variabele X worden toegekend. In het tweede voorbeeld zal de string aan de variabele X$ worden toegekend. Het intikken van een niet-numerieke waarde bij de bovenste INPUT opdracht zal leiden tot een foutmelding. Het intikken van een numerieke waarde bij de tweede INPUT opdracht zal ervoor zorgen dat de computer de numerieke waarde als string opvat.
INPUT kan niet direkt een waarde toekennen aan een array-variabele. daarvoor dient u een tussen-variabele te gebruiken.
Bijvoorbeeld:

```
INPUT X : ARRAY(1)=X
```

Het is niet mogelijk de INPUT-opdracht de ingetikte waarde direkt aan ARRAY(1) toe te laten kennen.
Als u een INPUT-opdracht gebruikt om een string aan een string-variabele toe te kennen, dient deze string eerder in het programma te zijn gedimensioneerd.
Het is mogelijk in een enkele INPUT-opdracht meer dan een waarde toe te kennen aan verschillende variabelen. Daarvoor worden de verschillende variabelen waaraan de waarden toegekend moeten worden, gescheiden door komma's.
Bijvoorbeeld:

```
INPUT X,Y,Z
```

of

```
INPUT X$,Y$,Z$
```

of

```
INPUT X,Y$,Z,A,B$
```

Bij elke invoer van waarden moet de juiste soort informatie opgegeven worden. Dat wil zeggen een string voor een stringvariabele en een numerieke waarde voor een numerieke variabele. De verschillende waarden kunnen bij het invoeren worden gescheiden door komma's of door het drukken op de RETURN toets. De computer zal net zo lang vraagtekens op het scherm blijven zetten, totdat aan alle INPUT opdrachten is voldaan.
Bij de invoer van stringgegevens kunt u de komma's niet gebruiken om de invoer te scheiden. De computer beschouwt de komma's als onderdeel van de string. Bij het invoeren van numerieke gegevens verwacht de computer enige invoer.
Als u alleen de RETURN toets indrukt, volgt een foutmelding. Dit geldt niet bij stringinvoer. Het enkel intikken van de RETURN toets wordt door de computer opgevat als het invoeren van een lege (nul-) string. Als u meer letters

invoert dan de lengte van de string toelaat (de lengte van de string moet eerst bepaald zijn door een DIM opdracht) zal de computer het teveel aan letters negeren. Deze worden niet in de string geplaatst.

Als u verder niets opgeeft, verwacht de INPUT opdracht de gegevens van het toetsenbord (IOCB kanaal nummer 0). De End-Of-Line opdracht (EOL) geeft aan dat de gegevens voor een enkele INPUT zijn voltooid. Bij invoer via het toetsenbord is de komma daarvoor bedoeld, of de RETURN-toets.

Voorbeelden:
```
INPUT Z
INPUT Q$
INPUT #4, TEST$
INPUT #2, G,H,P
```

Als u het toetsenbord als invoerapparaat gebruikt, kunt u beter geen gebruik maken van de cursortoetsen. Dit kan ertoe leiden dat het vraagteken van de INPUT opdracht deel uit gaat maken van de ingevoerde string (of van het ingevoerde numerieke gegeven, hetgeen een foutmelding oplevert).

Het gebruik van de INPUT opdracht voor de invoer van gegevens van andere bronnen dan het toetsenbord is voor alle apparaten hetzelfde. Bij elk ander apparaat dan het toetsenbord, fungeert het EOL karakter als het indrukken van de RETURN toets. Numerieke waarden kunnen door komma's gescheiden worden.
Meer gegevens proberen in te voeren dan het bestand lang is, levert een foutmelding op.
De algemene vorm van de INPUT opdracht bij gebruik van andere apparaten dan het toetsenbord is:

INPUT #kanaal, variabele

Bij deze opdracht waarbij #kanaal een geopend IOCB kanaal aangeeft, wordt er geen vraagteken op het scherm geplaatst.

Voorbeelden:
```
INPUT Z
INPUT Q$
INPUT #4,TEST$
INPUT #2, D,E,F
```

INT(x)

Een numerieke funktie die de grootste waarde kleiner dan x als resultaat geeft. x is een getal of een numerieke uitdrukking.
Bijvoorbeeld:

```
PRINT INT(5.3)      resultaat: 5
PRINT INT(5.999)    resultaat: 5
```

PRINT INT(-6.1)   resultaat: -7
PRINT INT(-6.9)   rseultaat: -7

Let op dat bij het gebruik van negatieve waarden, de INT opdracht ook de grootste waarde die kleiner is dan de opgegeven waarde als resultaat heeft.

Voorbeeld: X=INT(Q/3)

LEN (x$)

Een stringfunktie die als resultaat het aantal lettertekens in string x$ geeft.

Voorbeeld: A$=B$(LEN(B$)+1)

LET x=uitdrukking
LET x$=string

Door de LET statement wordt aan een variabele een waarde toegekend. De variabele kan zowel een string- als een numerieke variabele zijn. De toegekende waarde kan een string, een getal of een numerieke expressie zijn.

Voorbeelden: LET A=19
LET Q=Z
LET A$="STRING"

LIST "uitvoerapparaat" beginregel , eindregel

De LIST opdracht wordt gebruikt om informatie naar een uitvoerapparaat te sturen in ATASCII vorm. Als er geen uitvoerapparaat wordt opgegeven, gaat de computer ervan uit dat het beeldscherm wordt bedoeld.
Met 'beginregel' en 'eindregel' kan een deel van het programma worden gelist. Bijvoorbeeld:

LIST	list het gehele programma
LIST "P:"	list het gehele programma op de aangesloten printer
LIST 20	list alleen de opgegeven regel (20)
LIST 20-100	list vanaf de eerste opgegeven regel (20) tot de laatste opgegeven regel (100)
LIST "D:TEST.BAS"	list het programma op de diskette onder de naam 'TEST.BAS'

De LIST opdracht maakt alle gebruikte afkortingen in een programma (de BASIC statements) voluit. Rond de variabelen en statements worden, indien nodig, spaties geplaatst om de listing beter leesbaar te maken.
Een programmaregel kan maximaal 3 beeldschermregels in beslag nemen. De computer rekent eerst de gevolgen van het voluit schrijven van BASIC statements en het invoegen van spaties uit. Als er niet genoeg ruimte is, wordt

dit beter leesbaar maken van de listing achterwege gelaten. U kunt extra lange programmaregels maken door veel gebruik te maken van afkortingen en zo veel mogelijk spaties weg te laten. De regels zijn dan wel onhandelbaar en moeilijker te verbeteren. (Zie voor de toegestane afkortingen de tabellen aan het einde van deze bijlage.)
De LIST opdracht geeft de gegevens in ASCII vorm af, ongeacht het gebruikte uitvoerapparaat. Door middel van ENTER kan de informatie weer worden gelezen.
De LIST opdracht schrijft de tabel met variabelenamen niet weg naar het uitvoerapparaat.
De LIST opdracht gebruikt voor uitvoer naar andere apparaten altijd kanaal 7. Alleen uitvoer naar het beeldscherm gaat via kanaal 0. Kanaal 7 kan worden gebruikt door LIST ook al is dit kanaal al open. Na gebruik sluit LIST kanaal 7 altijd. De LIST opdracht sluit alle geluidsgeneratoren af.

De opdracht LIST "C:" werkt met de programmarecorder. U hoort twee piepjes om aan te geven dat u de cassette op de juist plaats moet zetten. Zet de programmarecorder op opnemen (PLAY en RECORD) en druk op een willekeurige toets (behalve BREAK) op het toetsenbord. De computer zal nu enig geluid maken. Zodra dit geluid ophoudt, is het programma op de cassette geplaatst. U ziet nu de READY mededeling.

Om LIST met een diskdrive te gebruiken, moet de DOS zich in het geheugen bevinden. Als u deze na het aanzetten van de computer niet geladen heeft, of er zijn fouten geweest bij het laden, zal de LIST"D:" opdracht tot een foutmelding leiden.

U kunt de LIST opdracht onderbreken door het indrukken van de BREAK toets of de SYSTEM RESET toets. Beide zorgen ervoor dat de uitvoer wordt stopgezet.
Een cassettebestand zal onvolledig zijn. U kunt dit onvolledige bestand toch lezen met een ENTER opdracht.
Een diskbestand kan helemaal niet op disk terecht gekomen zijn. Het is echter mogelijk dat de computer eerst het bestand afmaakt, en dan pas stopt met de LIST opdracht. In dat geval komt het gehele bestand op de diskette voor. Een ongelukkige timing kan ervoor zorgen dat de computer helemaal vastloopt door een SYSTEM RESET tijdens een LIST opdracht. Alleen het uit en waar aan zetten van de computer helpt dan nog.

LOAD invoerapparaat

LOAD wordt gebruikt om BASIC programma's of bestanden van een invoerapparaat in het geheugen van de computer te laden. Het opgegeven apparaat moet geschikt zijn voor het afgeven van de gewenste informatie en het programma of het bestand moeten opgeslagen zijn in gecodeerde vorm door een SAVE opdracht.
Programma's die weg geschreven zijn door middel van CSAVE of LIST kunnen niet door LOAD worden geladen.

De LOAD-opdracht gebruikt kanaal 7 (ook als dit kanaal al open is).
De LOAD-opdracht houdt automatisch een NEW opdracht in. Alle programmaregels en variabelen worden uit het geheugen gewist.
Bij het eindigen van de LOAD opdracht (doordat de opdracht volbracht is, of doordat de opdracht onderbroken is), worden de geluidsgeneratoren uit gezet en worden alle invoer/uitvoer kanalen (behalve kanaal 0) gesloten.

Bij gebruik van de programmarecorder zorgt de LOAD"C:" opdracht voor het laten klinken van een enkele piep. U kunt nu de cassette naar de juiste positie spoelen. Zet de programmarecorder op afspelen (PLAY) en druk op een willekeurige toets (behalve de BREAK) van het toetsenbord. De computer zal nu enige geluiden laten horen. Zodra deze ophouden, is de computer klaar met laden. De READY mededeling zal op het scherm verschijnen.

Om de LOAD"D:" opdracht te kunnen gebruiken moet de DOS zich al in het geheugen van de computer bevinden. U moet dit geladen hebben na het aanzetten van de computer. Als de DOS ontbreekt, of de diskdrive niet goed aangesloten is, of het opgegeven programma of bestand niet op de diskette voorkomt, volgt een foutmelding.

U kunt een LOAD opdracht alleen stoppen door een SYSTEM RESET. Dit kan tot gevolg hebben dat de computer 'hangt'. Dat wil zeggen dat de computer nergens meer op reageert. Het uit en weer aan zetten van de computer is de enige remedie.
Er zullen geen programmaregels in het geheugen voorkomen totdat de LOAD opdracht helemaal is afgewerkt.
Het indrukken van de BREAK toets zorgt er meestal alleen voor dat de LOAD opdracht even wordt onderbroken. Dit kan tot gevolg hebben dat een deel van de informatie verloren is.

Voorbeelden: LOAD "C:"
LOAD "D:PROGRAMMA.BAS"

LOCATE kolom, rij, x

Een grafische funktie die bepaalt welk teken of grafische code zich op de opgegeven schermpositie bevindt. De gevonden waarde wordt toegekend aan de variabele x. De schermpositie wordt opgegeven in kolom en rij. De opgegeven waarden moeten passen binnen de in gebruik zijnde schermmodus.
In schermmodus 0 geeft de LOCATE funktie de ATASCII waarde van het letterteken op positie kolom, rij.
In de andere tekstschermen zijn de verkregen waarden dezelfde als door de COLOR opdracht worden gebruikt. Zie onder COLOR.
In de grafische modi geeft de waarde aan welk kleurregister werd gebruikt voor het opgegeven punt.
De LOCATE-opdracht maakt gebruik van kanaal 6. Dit kanaal wordt automatisch geopend door een GRAPHICS opdracht.

Elke keer als de LOCATE-opdracht een waarde van het scherm leest, wordt de cursorpositie aangepast. Als u een LOCATE opdracht gebruikt nadat u de laatste positie van de laatste regel van het scherm heeft gelezen, volgt een foutmelding.

Door een PRINT-opdracht na een LOCATE-opdracht, kan het letterteken of het grafische punt waar de LOCATE-opdracht net geweest is, veranderd worden, waardoor de weergave wordt vernield. U kunt dit voorkomen door de POSITION opdracht te gebruiken om de cursor een plaats terug te plaatsen en vervolgens met de PUT-opdracht de net gevonden waarde weer terug te plaatsen.

Voorbeeld: LOCATE 5,10,X
LOCATE PEEK(86)*256+PEEK(85),PEEK(84),Q

LOG(x)

Een numerieke funktie die als resultaat heeft de natuurlijke logaritme (grondtal e) van x. x is een positief getal of een numerieke expressie groter dan nul.

Voorbeeld: X=Q*LOG(Z)

LPRINT expressie(s)

Deze opdracht is vergelijkbaar met de PRINT opdracht. In dit geval gaat de uitvoer niet naar het scherm maar naar de aangesloten printer.
Net als bij de PRINT opdracht kunnen achter PRINT allerlei uitdrukkingen worden geplaatst. De computer werkt de expressie(s) uit en drukt het resultaat af. Hoe het resultaat wordt afgedrukt, is afhankelijk van de expressie en van het gebruik van komma's en punt-komma's.
De LPRINT opdracht sluit alle geluidsgeneratoren af.

LPRINT zonder enige uitdrukking erbij levert een EOL (End of LINE) code af.
Numerieke waarden worden in de wetenschappelijke notatie afgedrukt als er meer dan 10 cijfers voor de komma staan, of als de waarde dichter bij nul ligt dan 0.01. Negatieve waarden worden voorafgegaan door een min-teken, positieve getallen worden zonder teken afgedrukt.
Een punt-komma tussen twee af te drukken waarden, zorgt ervoor dat de twee waarden zonder tussenruimte worden afgedrukt.
Een komma tussen twee af te drukken waarden, zorgt ervoor dat de tweede waarde in de volgende kolom wordt afgedrukt. Standaard zijn de kolommen 10 posities breed. Als een letterteken op een van de laatste twee posities van een kolom wordt afgedrukt, wordt de daarop volgende kolomstop uitgezet.

LPRINT maakt gebruik van kanaal 7 voor uitvoer naar de printer. Is dit kanaal al geopend, dan volgt een foutmelding. Door de foutmelding wordt het kanaal gesloten. Een tweede poging zal succesvol zijn.

Voorbeelden: LPRINT "WEEK NR. ";NR
LPRINT "Dit is een test-zin"

NEW

Door NEW worden alle BASIC programmaregels uit het geheugen gewist. Alle variabelen worden verwijderd, alle IOCB kanalen (behalve kanaal 0 voor uitvoer naar het beeldscherm) worden gesloten, de geluidsgeneratoren worden uitgezet en goniometrische funkties worden weer in radialen behandeld.

Voorbeeld: NEW

NEXT

De NEXT-statement is zinloos zonder een eerder gegeven FOR.. statement. Zie bij FOR.

NOTE #kanaal,sec,by

Deze statement kijkt waar de wijzer voor het diskbestand, geopend via het aangegeven bestand, naar wijst. De sector waar naar wordt gelezen, wordt opgeslagen in de variabele sec, de byte binnen de sector waar naar wordt verwezen, wordt opgeslagen in de variabele by. De variabelen mogen geen array-elementen zijn.
Het doet er niet toe voor welke taak het kanaal naar een diskbestand geopend is.
NOTE is niet beschikbaar onder DOS 1.0.

Voorbeelden: NOTE #4,SECTOR,BYTE
NOTE #2,3,78

ON x GOSUB regel1,regel2,....

Een voorwaardelijke sprong naar een van de subroutines van het programma. Het programma vertakt naar regel1 als de waarde van x gelijk is aan 1. Het programma vertakt naar regel2 als de waarde van x gelijk is aan 2. Het programma vertakt naar regel3 als de waarde van x gelijk is aan 3, etc.
Een RETURN opdracht stuurt aan het einde van de subroutine het programma terug naar de regel volgend op de regel waar de ON..GOSUB opdracht in staat.
De waarde van x moet liggen tussen 0 en 256. In andere gevallen gaat het programma verder met de regel volgend op de regel waar de ON..GOSUB opdracht in staat. De waarde van x moet een integer (een geheel getal) zijn. De regelnummers die aangegeven zijn door regel1, regel2, etc. moeten bestaande regelnummers of expressies, die leiden tot bestaande regelnummers in het programma, zijn. De regels hoeven niet in volgorde van regelnummer te staan.

Voorbeelden: ON X GOSUB 1000,1250,1500,750
ON Q GOSUB Q+100,Q*10,Q,2000

ON x GOTO regel1,regel2,...

Deze statement is vergelijkbaar met de ON..GOSUB statement. In dit geval wordt niet gesprongen naar een van de subroutines in het programma, maar naar een van de programmaregels van het programma. Het programma wordt bij de op deze wijze bereikte programmaregel voortgezet. Zie ook ON..GOSUB..

Voorbeelden: ON X GOTO 100,200,50,1000
ON Q GOTO Q,Q*10,Q*20,3000

OPEN #kanaal,taak,aux,"apparaat"

Voordat BASIC een extern apparaat kan gebruiken voor invoer of uitvoer, moet er eerst een kanaal naar dat apparaat worden geopend. Als u een kanaal probeert te openen dat al open is, volgt een foutmelding.
Veel BASIC-statements zorgen zelf voor het openen en sluiten van een kanaal.
Er moeten achter OPEN # vier waarden worden ingevuld:

- kanaal: deze waarde geeft het gebruikte kanaal aan. U kunt kiezen uit de kanalen 1 tot en met 7. Kanaal 0 wordt gebruikt voor invoer en uitvoer van het toetsenbord en naar het beeldscherm. Kanaal 6 wordt normaal gesproken gebruikt voor grafische bewerkingen.
- taak: deze waarde geeft door middel van een code aan voor welke taak het kanaal wordt geopend. Een kanaal dat geopend is voor invoer, kan niet worden gebruikt voor uitvoer en vice versa. Zie de tabel in de bijlage IOCB voor de mogelijke codes.
- aux: deze waarde wordt niet vaak gebruikt. Bijna altijd dient hier een 0 te worden ingevuld. Zie de bijlage IOCB voor de te gebruiken waarden.
- "apparaat": hier vult u de lettercode voor het te gebruiken apparaat in. Zie bijlage IOCB.

Voorbeelden: OPEN #1,4,0,"C:"
OPEN #5,8,0,"D:BESTAND.TXT"
OPEN #2,8,0,"P:"

OR

Een logische operator die wordt gebruikt in voorwaardelijke statements als IF...OR...THEN ..
Zie het hoofdstuk 'Logica en IF..THEN..'.

PADDLE(x)

PADDLE geeft als resultaat een geheel getal tussen 1 en 228. De waarde is afhankelijk van de stand van de regelbare weerstand (paddle) die aangesloten

is op paddle ingang x. De kleinste waarde duidt erop dat de paddle helemaal naar rechts (richting van klok) gedraaid is. De hoogste waarde duidt op een paddle de helemaal naar links (tegen de richting van de klok in) is gedraaid. Er is geen logisch verband tussen de hoek waaronder de paddle gedraaid is en de afgegeven waarde van deze statement.

Voorbeelden: PLOT PADDLE(0)/6
IF PADDLE(1)=BOLVER AND BOLHOR=180 THEN GOSUB 500

PEEK(adres)

PEEK is een speciale funktie die als resultaat de inhoud van een opgegeven geheugenadres geeft. Het resultaat is de decimale waarde van de op het adres opgeslagen byte.
Sommige gegevens vragen meer dan een enkel adres aan geheugenruimte. In zo'n geval is het nodig om meer dan een adres in het geheugen te bekijken met PEEK om de juiste waarde te vinden.
Bijvoorbeeld: Het beginadres, van dat deel van het geheugen waar de gegevens voor het beeldscherm in staan, staat in twee geheugenadressen (88 en 89). U vindt deze waarde door:

```
PRINT PEEK(88)+256*PEEK(89)
```

De waarde van adres 89 moet met 256 vermenigvuldigd worden om het juiste adres te vinden. De twee geheugenplaatsen zijn wel apart, maar hebben met elkaar te maken. Ze vormen samen een adres.

Voorbeeld: X=PEEK(Z*8)

PLOT x,y

Een grafische opdracht die een punt zet op de positie die aangegeven wordt door x,y. x staat daarbij voor het kolomnummer en y staat voor de rij. De minimumwaarden voor x en y zijn altijd 0. De maximum waarden zijn afhankelijk van de gebruikte schermmodus. Zie het hoofdstuk 'De schermen'. Het punt wordt getekend in de kleur die door de laatste COLOR-opdracht is gegeven. Is er nog geen COLOR-opdracht geweest sinds de laatste wisseling van schermmodus, dan wordt COLOR 0 (achtergrondkleur) genomen.

Voorbeelden: PLOT 10,19
PLOT X+12,SIN(Y)*30

POINT #kanaal,sec,by

De diskettebestand-wijzer van een via het IOCB kanaal (kanaal) geopende bestand, wordt verplaatst naar een bepaalde byte (by) van een bepaalde sector (sec).

De waarde van by moet liggen tussen 0 en 125.
Als de waarde van sec groter is dan de lengte van het bestand toelaat, volgt een foutmelding.
Het bestand moet geopend zijn naar een diskettebestand voor invoer, verbetering of toevoeging (zie bijlage IOCB).
POINT is het tegenovergestelde van NOTE.
Deze statement is niet aanwezig in DOS 1.0.

Voorbeelden: POINT #3,SECTOR,BYTE
POINT #2,1,100

POKE adres,geg

Een byte informatie (geg) wordt opgeslagen in het opgegeven geheugenadres (adres).
geg moet een waarde hebben tussen 0 en 255. Andere waarden kunnen niet in een enkele byte opgeslagen worden.
U kunt alleen naar RAM geheugenadressen POKEn. POKEt u naar een ROM adres of naar een geheugenadres dat niet in uw computer voorkomt, dan heeft de opdracht geen zin.
POKE is het tegenovergestelde van PEEK.
Let op! Met het POKEn kunt u uw computer niet beschadigen. Maar het is mogelijk dat u een waarde POKEt in een adres dat al in gebruik is door het bedrijfssysteem, BASIC, of een programma. In dat geval loopt u de kans uw programma te verminken, of de computer te laten 'hangen'. De enige remedie is het uit- en weer aanzetten van de computer.

Voorbeelden: POKE 756,226
POKE Z+1,Q*40

POP

Door POP vergeet de computer dat hij bezig is met een FOR..NEXT lus, een GOSUB lus of een ON..GOSUB lus.
Elke keer als de computer een FOR, GOSUB of ON..GOSUB-opdracht tegenkomt, wordt op een speciale plaats het regelnummer opgeslagen waar het programma weer terug moet komen na het uitvoeren van de NEXT of RETURN-opdracht.
Door de POP-opdracht wordt deze speciale waarde gewist.

Voorbeeld: POP

POSITION kolom,rij

De cursor wordt naar de opgegeven plaats op het scherm verplaatst.
De minimumwaarden voor kolom en rij zijn altijd 0. De maximumwaarden hangen af van de gebruikte schermmodus. Zie hoofdstuk 14.

Voorbeelden: POSITION 5,14
POSITION X,Y+5

PRINT expressielijst

PRINT voert lettertekens uit naar het beeldscherm.
Alleen PRINT stuurt een EOL (End of Line) teken naar het beeldscherm. Deze zorgt ervoor dat de cursor zich naar de eerste positie van de volgende regel begeeft.
Door achter PRINT een of meer uitdrukkingen te zetten, worden de resultaten van die expressies afgedrukt op het beeldscherm.
Door achter PRINT een IOCB kanaal op te geven (#kanaal), worden de resultaten niet naar het beeldscherm uitgevoerd, maar naar het kanaal.
De manier waarop de resultaten worden uitgevoerd door de PRINT-opdracht, heeft niets te maken met het uitvoerapparaat, maar alleen met het feit of het strings of numerieke gegevens zijn, en met het gebruik van komma's en punt-komma's.
Een komma tussen twee expressies zorgt ervoor dat het volgende item wordt afgedrukt in de volgende kolom. De kolom is normaal gesproken tien posities breed. Wordt er een letterteken in een van de twee laatste posities van een kolom afgedrukt, dan wordt de eerstvolgende kolomstop genegeerd.
Een punt-komma tussen twee items zorgt ervoor dat de twee items zonder enige tussenruimte afgedrukt worden.

Numerieke waarden met meer dan 10 cijfers voor de komma, of tussen 0 en 0.01, worden in de wetenschappelijke (E) notatie weergegeven. Negatieve getallen worden voorafgegaan door een minteken. Positieve waarden worden niet voorafgegaan door een teken.

PRINT gegevens gaan naar het beeldscherm als er geen uitvoerkanaal is opgegeven, of als het uitvoerorgaan is opgegeven als S: of E:.
Ongeacht de schermmodus geeft de PRINT-opdracht altijd lettertekens weer die voorkomen in de ATASCII lettertekenset (zie bijlage 1). In schermmodus 0 worden ze weergegeven zoals in de ATASCII lijst staat. In de schermmodi 1, 2, 12 en 13 worden ze vertaald in andere lettertekens. In de grafische schermen worden de PRINT gegevens vertaald in gekleurde punten.
Zie ook hoofdstuk 14.

De uitvoer van de PRINT-opdracht op het beeldscherm begint bij de huidige cursorpositie. Deze positie is opgeslagen in de geheugenlokaties 84 (rij) en 85 en 86 (kolom). De cursorpositie wordt beinvloed door de opdrachten DRAWTO, GET, INPUT, LOCATE, PLOT, POSITION, PRINT, PUT en XIO.

Bij het gebruik van de PRINT opdracht met een programmarecorder, moet een kanaal naar de recorder zijn geopend voor uitvoer (zie bijlage IOCB). De uitvoerbuffer verzamelt net zolang PRINT uitvoer tot de buffer vol is (128 bytes) en zendt dit dan in een keer naar de cassette. Een EOL (End Of Line) teken zorgt ervoor dat de buffer alles naar de cassette stuurt, of de buffer nu vol is

of niet. Als de buffer niet vol is, bevat de 128-ste byte als gegeven de lengte van het overgestuurde blok.
Als het uitvoerkanaal open is voor normale uitvoer, kan de programmarecorder tussen de uitvoer van de blokken stoppen. Als het uitvoerkanaal ingesteld is op een korte interval tussen de blokken, zal de recorder door blijven lopen. Uw programma moet dit tempo bijhouden, anders krijgt u onzin op uw cassette.

De PRINT-opdracht kan ook worden gebruikt voor uitvoer naar de diskette. Een bestand moet geopend zijn met een kanaal voor uitvoer, verbetering of aanvulling. In het algemeen zullen de gegevens op dezelfde wijze op de diskette worden geplaatst zoals zij op het scherm worden geplaatst.
De uitvoer gaat per blok. De buffer verzamelt gegevens tot een blok vol is en stuurt dit dan over. Een EOL teken zorgt ervoor dat een blok, vol of niet vol, wordt overgestuurd.

Om de PRINT-opdracht met een printer te gebruiken, moet een kanaal naar de printer geopend zijn, en een eventuele interface aangesloten zijn. De door printers gebruikte ASCII lijst wijkt meestal af van de ATASCII lijst.

Voorbeelden:

```
PRINT "Dit is een tekst"
PRINT "Score: ";SCORE
PRINT #6, "Y-as"
PRINT #2; X,X$,Y,Y$
```

PTRIG x

Deze funktie kijkt of de vuurknop van de paddle (x) wel of niet ingedrukt is: wel ingedrukt geeft een waarde van 0, niet ingedrukt geeft een waarde van 1

Voorbeeld: `IF PTRIG(1)=0 THEN PRINT "KNAL!"`

PUT #kanaal,x

De PUT-statement voert een enkele integer (geheel getal) waarde of expressie x, naar het opgegeven IOCB kanaal (#kanaal). Deze statement is het omgekeerde van de GET-statement.
De PUT-statement rondt de waarde van x af naar de dichtst bijzijnde integer. Als de waarde van x niet tussen 0 en 255 ligt, wordt er (een veelvoud van) 256 afgetrokken van x.
De PUT-statement doet hetzelfde als de PRINT-statement, maar dan voor een enkele integer waarde. Zie ook bij PRINT.

Voorbeelden:

```
PUT #1,10
PUT #KAN,X
```

RAD

Na deze opdracht beschouwt de computer de invoer bij goniometrische funkties als gegeven in radialen in plaats van graden. De tegenhanger van deze opdracht is de DEG-opdracht.
Door NEW, RUN of een SYSTEM RESET, kiest de computer automatisch voor radialen.

Voorbeeld: RAD

READ x,y,...

Door de READ statement worden waarden uit de DATAlijst toegekend aan de opgegeven variabelen. De computer heeft een wijzer naar de DATAlijst die bepaalt welk gegeven het eerstvolgende is om toe te kennen.
Bij het begin van het programma en na een RESTORE opdracht wijst deze wijzer naar het eerste gegeven in de eerste regel met een DATA opdracht. Zodra door READ een gegeven wordt toegekend aan een variabele, schuift de wijzer een plaats op naar het volgende gegeven.
Het type variabele moet overeenkomen met het gegeven in de DATAlijst. Een stringvariabele kan wel een numeriek gegeven bevatten (dat dan opgevat wordt als een string), maar een numerieke variabele kan geen string bevatten.
De DATAlijst mag zich overal in het programma bevinden.
De READ opdracht kan ook als direkte opdracht worden gegeven. Er moeten wel voldoende gegevens in de DATAlijst aanwezig zijn, anders volgt een foutmelding.

Voorbeelden: READ A$, B$, X, Y
FOR A=1 TO 20 : READ Q,R$: NEXT A

REM

Een niet uit te voeren statement die niets anders zegt dan dat de tekst die achter REM staat niet tot BASIC behoort, maar slechts aantekeningen van de programmeur bij het programma zijn.
REM-statements worden wel bij een listing weergegeven, maar bij het runnen van een programma genegeerd.
U kunt een REM-statement niet gebruiken aan het begin van een regel met meer dan een opdracht. Alles achter de REM-statement op een bepaalde regel wordt genegeerd, dit geldt ook voor eventuele opdrachten na een dubbele punt.

Voorbeelden: REM Subroutine tekenen
GRAPHICS 0 : REM TEKSTSCHERM

RESTORE regelnummer

Laat de datawijzer naar een ander gegeven wijzen.

De datawijzer bepaalt welk gegeven door de volgende READ-opdracht gelezen wordt. RESTORE zonder regelnummer er achter, zorgt ervoor dat de datawijzer naar het eerste gegeven van de eerste DATA-opdracht wijst. RESTORE gevolgd door een regelnummer zorgt ervoor dat de datawijzer naar het eerste gegeven van de DATA-opdracht op die regel wijst.

Voorbeelden: RESTORE
RESTORE 190

RETURN

Te gebruiken aan het einde van een subroutine om het programma weer terug te voeren naar de regel volgend op de regel waarin de GOSUB of ON ..GOSUB opdracht voorkomt.
Door een POP-opdracht wordt de huidige lus vergeten. De computer vervalt in de oudere lus, of gaat verder met het programma met de opdracht volgend op de RETURN opdracht.
Een RETURN zonder GOSUB, of meer RETURNs dan GOSUBs leidt tot een foutmelding.
Voorbeeld:-RETURN

RND(x)

Een numerieke funktie die een willekeurig getal tussen 0 (inclusief 0) en 1 (zonder 1) geeft. x is een 'dummy'. Dat wil zeggen dat u voor x elke numerieke waarde kunt invullen. Een 0 is de gebruikelijke waarde.
Om getallen groter dan 1 te krijgen moet u een vermenigvuldigingsfactor gebruiken. Om een willekeurig getal van 1 tot en met 6 te krijgen (ter simulatie van een dobbelsteen) gebruikt u:

```
DOBBEL = INT(RND(0)*6+1)
```

De INT funktie zorgt ervoor dat er een geheel getal uitkomt. De vermenigvuldigingsfactor zorgt voor getallen tussen 0 en 5, en +1 zorgt voor getallen tussen 1 en 6.

Voorbeeld: X=INT(RND(0)*79) : Y=INT(RND(0)*40) : PLOT X,Y

RUN invoerapparaat

Door RUN wordt er van de direkte modus (waarin u direkt opdrachten en funkties aan de computer kunt opgeven) overgeschakeld naar de programma modus (waarin de opdrachten en funkties in de vorm van een programma aan de computer worden gegeven).

Als u geen invoerapparaat aangeeft, wordt het programma in het geheugen geRUNd. Als u een invoerapparaat geeft, wordt eerst het programma van het invoerapparaat -geladen en vervolgens geRUNd.

Met RUN kunnen alleen gecodeerde BASIC programma's worden geladen (opgenomen met SAVE).
RUN gebruikt invoerkanaal 7 voor invoer vanaf de cassette of diskette. RUN werkt ook als kanaal 7 al open is. Na gebruik wordt kanaal 7 wel gesloten.

Bij gebruik met een cassetterecorder laat de RUN-opdracht de luidspreker een keer piepen. U kunt nu de cassette naar de juiste positie spoelen.
Druk vervolgens op de PLAY knop en druk op een willekeurige toets (behalve de BREAK) van het toetsenbord. U hoort nu een tijdje lang enkele geluiden van de computer. Zodra deze ophouden is het programma geladen. Het programma wordt nu automatisch geRUNd.

Om RUN met een diskettebestand te gebruiken moet DOS zich al in het geheugen bevinden. U moet dit, na het aanzetten van de computer, geladen hebben. Foutmeldingen kunnen ontstaan door het ontbreken van DOS of van het opgegeven programma.

Door RUN te onderbreken met SYSTEM RESET is het mogelijk dat er geen enkele programmaregel in het geheugen is geplaatst. Dit gebeurt pas als het laden afgemaakt is. Door op SYSTEM RESET te drukken terwijl een diskprogramma wordt geladen, kan de computer gaan 'hangen'. Uit- en weer aanzetten van de computer is de enige remedie.
Het indrukken van de BREAK toets stopt het programma, maar stopt slechts zelden het laden van het programma. U loopt wel de kans dat uw programma verminkt in het geheugen terecht komt.

Voorbeelden: RUN
RUN "C:"
RUN "D1:NAAM.BAS"

SAVE uitvoerapparaat

Door deze opdracht wordt een BASIC programma van het geheugen gekopieerd naar een uitvoerapparaat.
Normaal gesproken is het uitvoerapparaat een cassetterecorder (C:) of een diskdrive (D:programma-naam). Als u een ander uitvoerorgaan neemt, zoals het beeldscherm of de printer, krijgt u meestal onzin te zien. De SAVE opdracht codeert het BASIC programma.
Een programma dat door middel van SAVE is opgeslagen, kan alleen door LOAD of RUN weer geladen worden. Tijdens het opslaan, schrijft de SAVE opdracht ook de tabel met variabelenamen weg.

Kanaal 7 wordt voor uitvoer gebruikt. Het hindert niet als kanaal 7 al open is. Na beeindiging van de SAVE-opdracht wordt kanaal 7 gesloten. Alle geluidsgeneratoren worden uitgezet.

Bij gebruik met een cassetterecorder zorgt de SAVE opdracht eerst voor een dubbele piep. De gebruiker kan nu de cassette op de juiste positie plaatsen met

de FAST FORWARD en REWIND knop. Vervolgens moeten de REC en PLAY knop en een willekeurige toets (behalve de BREAK) van het toetsenbord worden ingedrukt. U hoort nu een tijdje verschillende geluiden door de luidspreker. Zodra deze zijn afgelopen, is het wegschrijven klaar. U krijgt nu de READY melding op het scherm.

Om de SAVE-opdracht met een diskdrive te kunnen gebruiken moet DOS zich reeds in het geheugen bevinden. U moet dit laden onmiddellijk na het aanzetten van de computer. Als DOS ontbreekt of de diskdrive niet goed aangesloten is volgt een foutmelding.
Als er al een bestand met de opgegeven naam op de diskette staat, wordt het oude bestand vervangen door het nieuwe bestand. U kunt de werking van de SAVE-statement stoppen door de BREAK toets of de SYSTEM RESET toets in te drukken. In beide gevallen heeft u een niet volledig programma op cassette of diskette. Zo'n onvolledig programma kan niet worden gelezen door LOAD of RUN.

Voorbeelden: SAVE "C:"
SAVE "D:PROGRAMMA35"

SETCOLOR reg,kl,hel

Deze opdracht kent een bepaalde kleur (kl) in een bepaalde helderheid (hel) toe aan een kleurregister (reg).
Er zijn vijf verschillende kleurregisters: 0, 1, 2, 3 en 4. reg kan een van deze vijf waarden hebben.
kl geeft de kleur aan. Er zijn 16 kleuren mogelijk. De kleuren zijn genummerd van 0 tot en met 1. Zie bij COLOR voor de mogelijke kleuren en codes.
hel geeft de helderheid aan. U kunt hiervoor een geheel getal tussen 0 en 15 opgeven. Alleen de even waarden werken. De oneven waarden worden afgerond naar een even waarde, die kleiner is dan de opgegeven waarde.
Als u geen geheel getal opgeeft voor kl of hel, wordt de waarde afgerond naar het dichtst bijzijnde gehele getal. Van waarden groter dan 15 (tot een maximum van 65535) wordt (een veelvoud van) 16 afgetrokken om de waarde toch tussen 0 en 15 te laten vallen. Zie ook bij COLOR.

Voorbeelden: SETCOLOR 2,0,0
FOR A=0 TO 15 : SETCOLOR 1,5,A : NEXT A

SGN(x)

Een numerieke funktie die als resultaat 1, 0 of 1 heeft, naar gelang het teken van de variabele x. x moet een getal of een numerieke expressie zijn.

SGN(x) geeft -1 als x negatief is
SGN(x) geeft 0 als x nul is
SGN(x) geeft 1 als x positief is

Voorbeeld: IF SGN(Q)=-1 THEN PRINT "Q is negatief"

SIN(x)

Een numerieke funktie die de sinus van de hoek x opgegeven in radialen geeft. Als eerder de opdracht DEG gegeven is, gaat de computer ervan uit dat x opgegeven is in graden.

Voorbeeld: X=SIN(HOEK)

SOUND kanaal,hoogte,vorm,volume

Zet een geluidsgenerator aan of uit. Stelt de toonhoogte, de vorm van het geluid en het volume in.
De SOUND opdracht heeft betrekking op vier verschillende geluidsgeneratoren. Voor kanaal kunt u een getal tussen 0 en 3 invullen. De waarde die u voor hoogte invult, bepaalt de hoogte van de door de geluidsgenerator voortgebrachte toon. Zie hoofdstuk 20.
De waarde van vorm bepaalt de vorm van het geluid. Dit moet een even getal tussen 0 en 14 zijn. De waarden 10 en 14 geven een min of meer zuivere toon. Andere waarden geven vervormde geluiden. Zie ook hoofdstuk 20.
De waarde van volume bepaalt hoe hard de toon klinkt. Dit is mede afhankelijk van de volumeafstelling van uw televisie.
Alle vier de geluidsgeneratoren worden afgezet door het indrukken van SYSTEM RESET of door het uitvoeren van een van de volgende opdrachten: CLOAD, CSAVE, DOS, END, ENTER, LIST (behalve een LIST op het scherm), LOAD, NEW, RUN en SAVE.

Voorbeelden: SOUND 2,100,10,12
FOR A=1 TO 100:SOUND 0,A,14,6 : NEXT A

SQR(x)

Een numerieke funktie die de vierkantswortel van x geeft. x moet een positief getal of expressie zijn.

Voorbeeld: DIAG=SQR(LIJN1*LIJN1 + LIJN2*LIJN2)

STATUS #kanaal,x

Geeft de toestand van de laatste invoer of uitvoer bewerking op het aangegeven kanaal aan. De gegeven code wordt toegekend aan de variabele x. Als de code 128 of hoger is, is er iets fout gegaan. De codes met de betekenis zijn:

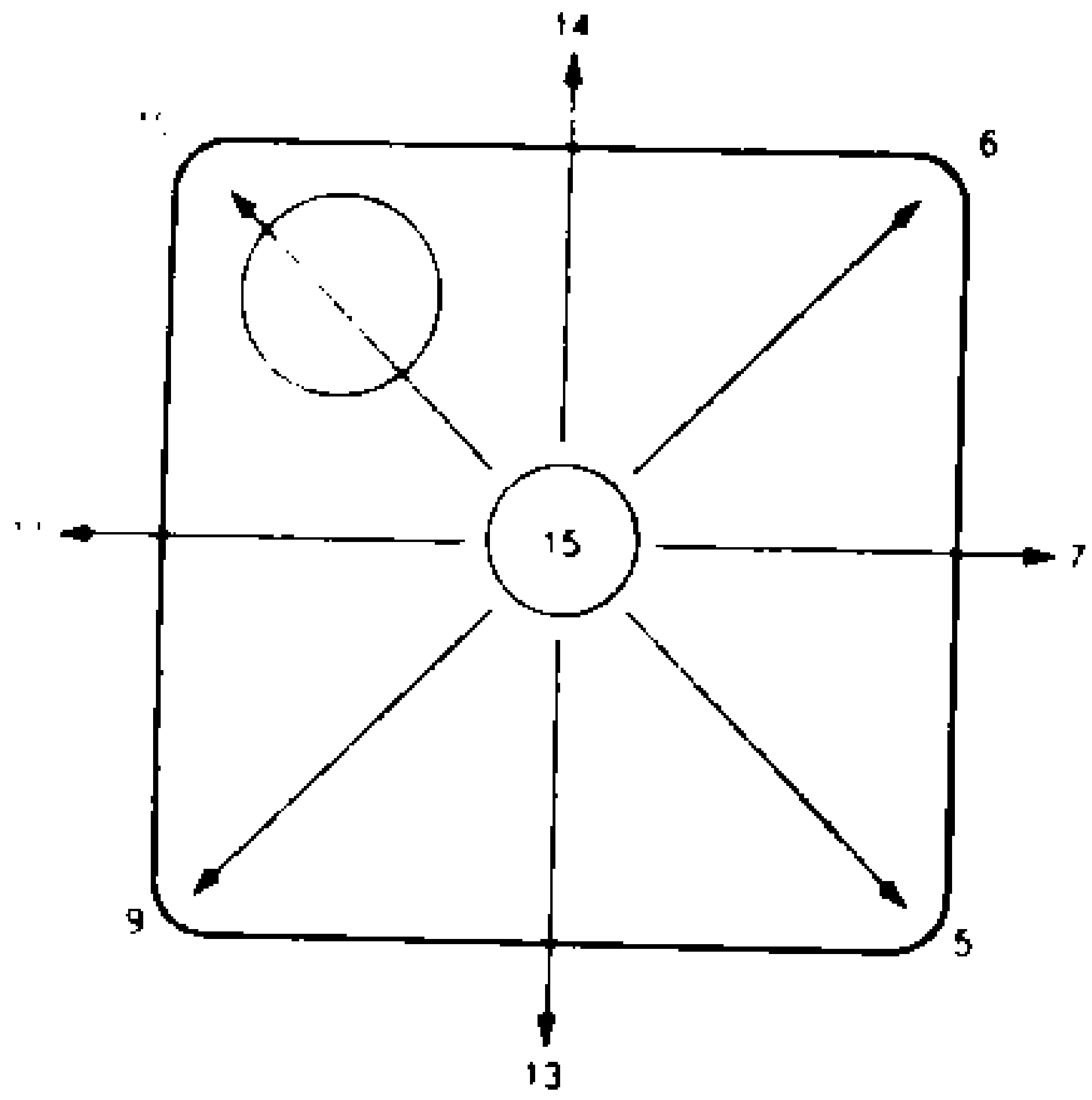
STATUS codes	
Code	Betekenis
001	bewerking klaar (geen fouten)
003	einde van bestand (EOF)
128	BREAK uitgevoerd
129	IOCB kanaal was al open
130	apparaat bestaat niet
131	kanaal alleen geopend voor schrijven
132	niet bestaande opdracht
133	apparaat of bestand niet open
134	ongeldig IOCB kanaal nummer
135	kanaal alleen geopend voor lezen
136	einde van bestand (EOF) tegen gekomen
137	record afgebroken
138	apparaat antwoordt niet
139	apparaat NAK
140	fout bij seriele bus
141	cursor buiten toegestane bereik
142	fout bij seriele bus
143	controle van seriele bus mislukt
144	schrijven naar diskette lukt niet
145	verkeerde beeldschermmodus
146	funktie niet bekend
147	niet genoeg geheugen voor scherm
160	diskdrive bestaat niet
161	teveel diskbestanden open
162	diskette vol
163	fatale diskette invoer/uitvoer fout
164	informatie op schijf verminkt
165	verkeerde bestandnaam
166	lengte van POINT is te lang
167	bestand op slot
168	opdracht kan niet bij diskette
169	inhoudsopgave van diskette is vol
170	bestand niet gevonden
171	POINT ongeldig

STEP

STEP is een niet verplicht onderdeel van de FOR..NEXT opdracht. **Zie bij** FOR.

STICK(x)

Geeft de stand van de joystick aan. Deze opdracht kan acht verschillende gehele getallen als resultaat hebben:



De joysticks zijn van 0 tot en met 3 genummerd. Als het nummer dat wordt opgegeven voor de joystick kleiner dan 0 of groter dan 255 is, volgt een foutmelding. Bij nummers tussen 4 en 255 volgen niet voorspelbare getallen.

ingang nummer	STICK(nr)
1	0
2	1
3	2
4	3

Voorbeeld: IF STICK(0)=14 THEN RIJ=RIJ-1

STRIG(x)

Deze funktie bepaalt of de vuurknop van de opgegeven joystick is ingedrukt. De funktie geeft als resultaat een 0 als de vuurknop is ingedrukt , en een 1 als deze niet ingedrukt is. Zie onder STICK voor de samenhang tussen ingang en STRIG nummer.

Voorbeeld: IF STRIG(1)=0 THEN PRINT "plof"

STOP

Laat een BASIC programma stoppen.
De computer keert terug naar schermmodus 0 en geeft als mededeling:

STOPPED AT LINE regelnummer

Het regelnummer geeft aan waar de computer is gestopt met het programma. Een CONT-opdracht laat de computer verder gaan met de uitvoering van het programma. Het programma wordt voort gezet vanaf de regel volgend op de regel waar het programma gestopt is.
STOP sluit de geluidsgeneratoren en de IOCB kanalen niet.

Voorbeeld: STOP

STR$(x)

STR$ zet een numerieke waarde x om in een string.

Voorbeeld: NUMMER$ = STR$(19+87)

THEN

THEN is een vast onderdeel van de IF..THEN statement. Zie bij IF.

TO

TO is een vast onderdeel van de FOR..NEXT lus. Zie bij FOR.

TRAP x

Een opdracht die de computer naar een opgegeven regelnummer x laat vertakken als er een fout wordt gevonden.
Door deze opdracht wordt het automatische fout-afhandelingssysteem uitgezet. Normaal springt de computer bij het vinden van een fout naar schermmodus 0, drukt een foutboodschap af en stopt het programma. Na de TRAP opdracht springt de computer alleen maar naar de opgegeven regel.
Elke fout heeft zijn eigen code. Zie voor de codes bijlage 1. De code van de laatste fout staat in geheugenadres 195. PEEK(195) geeft u de foutcode. In de geheugenadressen 186 en 187 vindt u het regelnummer waar de fout plaatsvond. 256*PEEK(187)+PEEK(186) geeft dit regelnummer als resultaat.
Een TRAP statement moet zodanig in het programma geplaatst worden, dat de computer de TRAP tegenkomt voordat er een fout wordt gevonden.
U kunt de TRAP mooi gebruiken voor het opvangen van fouten van gebruikers van uw programma. Een TRAP vlak voor een INPUT opdracht waar een getal gevraagd wordt, kan ervoor zorgen, dat als de gebruiker per ongeluk een letter invult, het programma niet stopt, maar naar een routine springt. Deze routine wijst de gebruiker er op dat hij/zij iets verkeerd intikt. Vervolgens springt het programma weer terug naar de oorspronkelijke INPUT opdracht om de gebruiker een nieuwe kans te geven. De computer moet daarbij weer langs de TRAP-opdracht komen, zodat ook een tweede vergissing wordt opgevangen.

Voorbeeld: TRAP 2000

USR(x)

Een funktie die de computer uit BASIC naar een machinetaal subroutine stuurt, die begint op het adres x.
Als de machinetaal subroutine zonder fouten is, keert de computer na de subroutine terug naar BASIC. Het BASIC programma wordt hervat op de regel volgend op de regel waarin de USR-statement staat.
Achter het adres kunnen, van rechts naar links, gegevens worden geplaatst, die door USR op de hardware stapel worden geplaatst.

Voorbeeld: A = USR(1427,128,ADR(M$))

VAL(x$)

Deze funktie zet een string om in een numerieke waarde.
Het eerste letterteken van de string moet een spatie, plus- of minteken of een getal zijn. In andere gevallen volgt een foutmelding.
Deze opdracht is het omgekeerde van STR$(x).

Voorbeeld: GETAL = VAL(A$) + VAL(B$)

XIO bew ,#kanaal ,para1 ,para2, apparaat

Dit is de algemene invoer/uitvoer opdracht van de Atari. De eerste waarde achter XIO (bew) bepaalt de bewerking. Zie tabel hieronder.
De tweede waarde geeft aan welk kanaal voor de XIO bewerking is geopend.
De laatste waarde geeft het te gebruiken apparaat aan. De twee parameters er tussen geven extra informatie die voor sommige bewerkingen nodig is. Deze worden niet altijd gebruikt, maar moeten wel altijd worden ingevuld.

XIO bewerkingen				
Werking	code	gelijk aan BASIC	para1	para2
open kanaal	3	OPEN	tabel	tabel
lees regel	5	INPUT	0	0
neem teken	7	GET	0	0
schrijf regel	9	PRINT	0	0
plaats teken	11	PUT	0	0
sluit kanaal	12	CLOSE	0	0
status kanaal	13	STATUS	0	0
trek lijn	17	DRAW-TO	0	0
vul gebied	18	geen	0	0
nieuwe naam bestand	32	DOS menu	0	0
wis bestand	33	DOS menu	0	0
sluit bestand	35	DOS menu	0	0
bestand van slot	36	DOS menu	0	0
verplaats wijzer	37	POINT	0	0
zoek bestand wijzer	38	NOTE	0	0
formateer diskette	254	DOS menu	0	0

BIJLAGE 3B: ATARI BASIC STATEMENTS

Naar werking, met afkortingen

BASIC COMMANDO'S	
COMMANDO	AFKORTING
BYE	B.
CONT	CON.
END	geen
LET	LE.
LIST	L.
NEW	geen
REM	R. of punt gevolgd door spatie
RUN	RU.
STOP	STO.

BASIC STUUR STATEMENTS	
STATEMENT	AFKORTING
FOR	F.
GOSUB	GOS.
GOTO	G.
IF	geen
NEXT	N.
ON	geen
POP	geen
RESTORE	RES.
RETURN	RET.
STEP	geen
THEN	geen
TO	geen
TRAP	T.

INVOER/UITVOER STATEMENTS	
STATEMENT	AFKORTING
CLOAD	CLOA.
CLOSE	CL.
CSAVE	CS.
DATA	D.
DOS	DO.
ENTER	E.
GET	GE.
INPUT	I.
LOAD	LO.
LPRINT	LP.
NOTE	NO.
OPEN	O.
PADDLE	geen
POINT	PO.
PRINT	PR. of ?
PTRIG	geen
PUT	PU.
READ	REA.
SAVE	S.
SOUND	SO.
STATUS	ST.
STICK	geen
STRIG	geen
XIO	X.

STRING STATEMENTS	
STATEMENT	AFKORTING
ASC	geen
CHR$	geen
LEN	geen
STR$	geen
VAL	geen

GRAFISCHE STATEMENTS	
STATEMENT	AFKORTING
GRAPHICS	GR.
COLOR	C.
DRAWTO	DR.
GET	GE.
LOCATE	LOC.
PLOT	PL.
POSITION	POS.
PUT	PU.
SETCOLOR	SE.
XIO	X.

ARRAY STATEMENTS	
STATEMENT	AFKORTING
DIM	DI.
CLR	geen
COM	geen

WISKUNDIGE EN DIVERSE FUNKTIES	
FUNKTIE	AFKORTING
ABS	geen
ADR	geen
ATN	geen
CLOG	geen
COS	geen
DEG	geen
EXP	geen
FRE	geen
INT	geen
LOG	geen
PEEK	geen
POKE	geen
RAD	geen
RND	geen
SGN	geen
SIN	geen
SQR	geen
USR	geen

BIJLAGE 4: BELANGRIJKE PEEK- EN POKE-ADRESSEN

Het beschrijven van alle geheugenadressen en hun funktie binnen de Atari zou een compleet boek vullen. In deze bijlage vindt u de adressen die voor een BASIC programmeur van belang kunnen zijn.

De inhoud van een geheugenlokatie wordt bekeken met de opdracht:

PEEK (adres)

De inhoud van een geheugenlokatie kan worden veranderd door er de gewenste waarde in te plaatsen. U doet dit met de opdracht:

POKE adres, waarde

WEERGAVE

14.15	Onderkant van display screen. In deze lokaties vindt u de hoogste lokatie die beschikbaar is voor het programma en de variabelen. Het geheugen gebied erboven wordt gebruikt voor het beeld scherm.
77	Als u langer dan negen minuten geen enkele toets indrukt, gaan de kleuren op het beeldscherm knipperen. Dit gebeurt om inbranden van de kleuren op uw kleurentelevisie te voorkomen. Een 0 in deze lokatie zorgt ervoor dat er niet meer wordt geknipperd.– Elke keer als u een toets indrukt wordt er automatisch een 0 in lokatie 77 geplaatst. De waarde 254 zet het knipperen in werking.
82	Linker kantlijn van tekst. Standaard staat hier 2. Er is een kantlijn van twee posities. Minimum in te vullen waarde is 0. Maximum in te vullen waarde is 39.
83	Rechter kantlijn van tekst. Standaard staat hier 39. Minimum waarde is 0, maximum waarde is 39. De waarde in 83 kan niet kleiner zijn dan de waarde in 82.
84	Cursorpositie (rij). Geeft de rij aan waar de cursor staat. Minimum waarde 0, maximum waarde afhankelijk van scherm.
85, 86	Cursorpositie (kolom). Geeft de kolom aan waar de cursor staat. PEEK(85) is minstens 0. De maximum waarde is afhankelijk van de schermmodus. De waarde in 86 is in de schermen 0 tot en met 7 altijd 0.
87	Bevat de huidige schermmodus.
88, 89	Laagste adres van het schermgeheugen. Het teken dat zich in dat adres bevindt, wordt linksboven op het scherm geplaatst.
90	Beginrij van grafische cursor (voor DRAWTO en XIO 18).
91, 92	Beginkolom van grafische cursor (voor DRAWTO en XIO 18).

93 Tijdelijke opslagruimte voor teken dat zich (onzichtbaar) onder de cursor bevindt. Als de cursor naar de volgende positie gaat, moet dit teken weer zichtbaar worden.

96 Eindrij voor grafische DRAWTO of XIO 18 opdracht.

97, 98 Eindkolom voor grafische DRAWTO of XIO 18 opdracht.

201 Aantal kolommen tussen de TAB stops. De eerste tabulatie gaat naar kolom nummer PEEK(201). De standaard instelling is 10.

656 Rijnummer van tekstcursor in tekstvenster. De waarde kan van 0 tot en met 3 lopen.

657,658 Kolomnummer van tekstcursor in tekstvenster.

752 Cursor aan/uit. Met een waarde 0 is de cursor zichtbaar, met een waarde 1 is de cursor onzichtbaar.

755 Besturingtekens en cursor volgens onderstaande tabel.

decimale waarde	kleur	Cursor aan/afwezig	tekens
0	transparant	afwezig	normaal
1	invers	afwezig	normaal
2	transparant	aanwezig	normaal
3	invers	aanwezig	normaal
4	transparant	afwezig	omgekeerd
5	invers	afwezig	omgekeerd
6	transparant	aanwezig	omgekeerd
7	invers	aanwezig	omgekeerd

756 Bepaalt welke lettertekens worden gebruikt in scherm modi 1 en 2. 224 geeft hoofdletters en cijfers, 226 geeft kleine letters en grafische tekens. Zie ook hoofdstuk 14.

763 ATASCII code van laatste gezette teken of waarde van laatste geplaatste grafische punt. Wordt gebruikt door de DRAWTO en XIO 18 opdracht.

PLAYER MISSILE GRAFIEK

559 Resolutie. 62 voor enkele lijn resolutie, 46 voor dubbele lijn resolutie.

623	Voorrangsregeling
	1: alle pm spelers voorrang boven normale grafiek 4: alle kleurregisters voorrang boven pm spelers 2: spelers 0 & 1, kleurregisters, spelers 2 & 3 8: kleurregisters 0 & 1, pm spelers, registers 2 & 3
704	kleur van speler + projectiel 0
705	kleur van speler + projectiel 1
706	kleur van speler + projectiel 2
707	kleur van speler + projectiel 3
53248	horizontale positie van speler 0
53249	horizontale positie van speler 1
53250	horizontale positie van speler 2
53251	horizontale positie van speler 3
53252	horizontale positie van projectiel 0
53253	horizontale positie van projectiel 1
53254	horizontale positie van projectiel 2
53255	horizontale positie van projectiel 3
53256	formaat speler 0 (0=normaal, 1=dubbel, 2=vier-dubbel)
53257	formaat speler 1 ..
53258	formaat speler 2 ..
53259	formaat speler 3 ..
53277	Een 3 zet de pm grafiek aan. Een 0 zet dit uit.
54279	Hoge byte van pmbegin moet hier geplaatst worden.

DIVERSEN

694	Inverse aan/uit. Een 0 geeft normale lettertekens. Een waarde groter dan 0 geeft inverse tekens.
764	Waarde van de laatst ingedrukte toets of de waarde 255 als er geen toets is ingedrukt.
65	Geluid tijdens invoer/uitvoer. Een 0 zorgt voor afwezigheid van geluid. Een waarde groter dan 0 zorgt voor geluid (dit is de standaard instelling)
195	Foutmelding. In deze lokatie staat de code van de geconstateerde fout.
20,19 18	Televisiebeeldentellers (50* per seconde) zie hoofstuk 7.

BIJLAGE 5: IOCB

Voor een uitleg over de verschillende opdrachten die gebruik maken van IOCB kanalen wordt u verwezen naar hoofdstuk 4, hoofdstuk 12, hoofdstuk 18 en bijlage 3.

IOCB kanalen

Kanaal	Funktie
0	altijd open voor scherm (E:)
1	vrij
2	vrij
3	vrij
4	vrij
5	vrij
6	wordt automatisch geopend en gesloten voor grafische bewerkingen
7	wordt automatisch geopend en gesloten voor invoer/uitvoer bewerkingen met een programmarecorder of diskdrive.

APPARAAT CODE'S

C:	Programmarecorder
D:	Diskdrive. Achter D: moet een bestandsnaam worden opgegeven. Achter de bestandsnaam mag een uitbreiding die het soort bestand aangeeft worden opgegeven. Bijvoorbeeld D:PROGRAMMA.BAS D:TEKST.TXT De bestandsnaam mag maximaal 8 letters of cijfers bevatten en moet beginnen met een letter. De uitbreiding moet beginnen met een punt, welke moet worden gevolgd door drie letters of cijfers.
D1: - D4:	Bij gebruik van meer dan een diskdrive, kunnen deze worden genummerd. Om D: te kunnen gebruiken moet DOS is het geheugen aanwezig zijn.
E:	Scherm opmaak
K:	Toetsenbord
P:	Printer
R:	RS-232 seriele poort. Het seriele poort programma moet in het geheugen aanwezig zijn voordat R: gebruikt kan worden.
R1: - R4	Bij gebruik van meer dan een seriele poort, kunnen deze genummerd worden.
S:	Beeldscherm

IOCB TAAK CODE'S

De volgende code's moeten bij samengesteld gebruik worden opgeteld:

1	aanvullen
2	disk bewerking
4	invoer
8	uitvoer
16	tekst-scherm vlag
32	scherm niet wissen

GEBRUIKELIJKE TAAK CODE'S

TAAK CODE	BEWERKING
	programmarecorder:
4	lezen vanaf de programmarecorder
8	schrijven naar de programmarecorder
	diskdrive
4	lezen vanaf de diskette
6	lezen van de inhoudsopgave
8	nieuw bestand wegschrijven
9	aanvullen van diskettebestand
	scherm-opmaak
8	schrijf naar scherm
9	anvullen naar scherm
12	toetsenbord invoer, scherm uitvoer
13	scherm invoer en uitvoer
	toetsenbord
4	lezen van toetsenbord
	printer
8	schrijven naar printer
	RS-232
4	lees blo
5	simultaan lezen

8	schrijf compleet blok
9	simultaan schrijven
13	simultaan lezen en schrijven

scherm

8	wis het scherm, geen tekstvenster alleen schrijven
12	wis het scherm, geen tekstvenster lezen en schrijven
24	wis het scherm, tekstvenster alleen schrijven
28	wis het scherm, tekstvenster lezen en schrijven
40	wis scherm niet (wel in modus 0) geen tekstvenster, alleen schrijven
44	wis scherm niet (wel in modus 0) geen tekstvenster, lezen en schrijven
56	wis scherm niet (wel in modus 0) tekstvenster, alleen schrijven
60	wis scherm niet (wel in modus 0) tekstvenster, lezen en schrijven

De AUX parameter bij de OPEN-opdracht is bijna altijd 0. Bij verschillende IOCB bewerkingen heeft de parameter echter wel nut.

AUX PARAMETER BIJ OPEN STATEMENT

Programmarecorder:	0	normale intervallen tussen blokken
	128	korte intervallen tussen blokken
Diskdrive	altijd 0	
Scherm opmaak	altijd 0	
Toetsenbord	altijd 0	
Atari printer	0	normale weergave
	83	zijdelings drukken (alleen Atari 820)
RS-232	altijd 0	
Beeldscherm	0	schermmodus 0
	1	schermmodus 1
	2	schermmodus 2
	3	schermmodus 3
	4	schermmodus 4
	5	schermmodus 5
	6	schermmodus 6
	7	schermmodus 7
	8	schermmodus 8

De XIO-opdracht is zeer veelzijdig. De eerste parameter bepaalt waarvoor de XIO-opdracht geschikt is.

XIO OPDRACHTEN

Algemene taken:

3	open een kanaal	OPEN
5	voer een regel in	INPUT
7	voer een teken in	GET
9	voer een regel uit	PRINT
11	voer een teken uit	PUT
12	sluit een kanaal	CLOSE
13	huidige IOCB status	STATUS
17	trek een lijn	DRAWTO
18	vul een gebied	-

Diskette taken:

32	herbenoem bestand	DOS menu
33	wis bestand	DOS menu
35	sluit bestand af	DOS menu
36	open bestand	DOS menu
37	verplaats bestand wijzer	POINT
38	zoek bestandwijzer	NOTE
254	formateer diskette	DOS menu

RS-232 taken:

32	voer kort blok uit
34	stel uitgaande lijnen in DTR, RTS en XMT
36	stel in: baud rate, woordlengte, stop bits
38	vertaal modi en parity
40	begin simultaan I/O modus

Bij alle XIO opdrachten moeten de derde en vierde parameter op 0 worden ingesteld. Bij de RS-232 bewerkingen hebben deze parameters echter wel zin.

XIO 34: PARAMETER 3 (parameter 4 moet op 0 staan)

Parameter	**DTR**	**RTS**	**XMT**
162	uit	uit	uit
163	uit	uit	aan
178	uit	aan	uit
179	uit	aan	aan
226	aan	uit	uit
227	aan	uit	aan
242	aan	aan	uit
243	aan	aan	aan

XIO 36: PARAMETERS 3 en 4

Parameter 3 waarde	funktie
0	1 stopbit woordlengte 8 baudrate 300
1	baudrate 45.5
2	baudrate 50
3	baudrate 56.875
4	baudrate 75
5	baudrate 110
6	baudrate 134.5
7	baudrate 150
8	baudrate 300
9	baudrate 600
10	baudrate 1200
11	baudrate 1800
12	baudrate 2400
13	baudrate 4800
14	baudrate 9600
15	baudrate 9600
16	woordlengte 7
32	woordlengte 32
48	woordlengte 48
128	2 stopbits

parameter 4			
waarde	DSR	CTS	CRX
0	nee	nee	nee
1	nee	nee	ja
2	nee	ja	nee
3	nee	ja	ja
4	ja	nee	nee
5	ja	nee	ja
7	ja	ja	ja

MIO 38: PARAMETER 3 (parameter 4 moet op 0 ingesteld zin)

LINE FEED		TRANSLATION		INVOER PARITY		UITVOER PARITY	
nee	0	licht	0	negeren	0	gelijk	0
ja	64	zwaar	16	oneven	4	oneven	1
		geen	32	even	8	even	2
				negeren	12	bit aan	3

BIJLAGE 6: DISK OPERATION SYSTEM

Het opstarten van de diskdrive:
- zet diskdrive aan
- wacht tot rode lampje vlak onder gleuf uit gaat
- open drive 'deur' door hendel horizontaal te draaien
- plaats diskette met etiket naar boven en aan handzijde in de drive
- sluit deurtje (hendel vertikaal) - zet computer aan

Door DOS te tikken, gevolgd door RETURN krijgt u het DOS menu. Op het ogenblik zijn er vier verschillende versies DOS op de markt. Het menu van elk van deze is:

DOS versie 1.0

DISK OPERATING SYSTEM 9/24/79
COPYRIGHT 1979 ATARI

A. DISK DIRECTORY	I. FORMAT DISK
B. RUN CARTRIDGE	J. DUPLICATE DISK
C. COPY FILE	K. BINARY SAVE
D. DELETE FILE	L. BINARY LOAD
E. RENAME FILE	M. RUN AT ADDRESS
F. LOCK FILE	N. DEFINE DEVICE
G. UNLOCK FILE	O. DUPLICATE FILE
H. WRITE DOS FILE	

DOS versie 2.0

DISK OPERATION SYSTEM II VERSION 2.0S
COPYRIGHT 1980 ATARI

A. DISK DIRECTORY	I. FORMAT DISK
B. RUN CARTRIDGE	J. DUPLICATE DISK
C. COPY FILE	K. BINARY SAVE
D. DELETE FILE(S)	L. BINARY LOAD
E. RENAME FILE	M. RUN AT ADDRESS
F. LOCK FILE	N. CREATE MEM. SAVE
G. UNLOCK FILES	O. DUPLICATE FILE
H. WRITE DOS FILES	

DOS versie 3.0

Atari DOS 3 Copyright 1983

File index	Load	Mem save
To cartridge	Save	Go at
Copy/append	Erase	Hex addr
Duplicate	Rename	X-user-
Init disk	Protect	defined
Access DOS 2	Unprotect	Help

DOS versie 2.5

DISK OPERATING SYSTEM II VERSION 2.5
COPYRIGHT 1984 ATARI CORP.

A. DISK DIRECTORY	I. FORMAT DISK
B. RUN CARTRIDGE	J. DUPLICATE DISK
C. COPY FILE	K. BINARY SAVE
D. DELETE FILE(S)	L. BINARY SAVE
E. RENAME FILE	M. RUN AT ADDRESS
F. LOCK FILE	N. CREATE MEM.SAVE
G. UNLOCK FILE	O. DUPLICATE FILE
H. WRITE DOS FILES	P. FORMAT SINGLE

De benamingen van de verschillende mogelijkheden van de menu's zijn niet altijd gelijk, maar de werking komt vaak overeen.
Bij DOS 1.0, 2.0 en 2.5 wordt een bepaalde funktie uit het menu gekozen door de letter die ervoor staat in te tikken. Bij DOS 3.0 wordt een funktie gekozen door de eerste letter (die inverse is weergegeven) in te tikken. In alle gevallen moet daarna de RETURN toets worden ingedrukt.
Nadat u een keuze hebt gemaakt, volgt meestal het verzoek om aanvullende informatie. U kunt dan nog eenvoudig terug naar het hoofdmenu. Als u in plaats van de gevraagde informatie, slechts RETURN indrukt, krijgt u het menu weer te zien.

U kunt elk item gebruiken bij elke diskette die u in uw diskdrive gebruikt. Voor bijna alle items moeten de diskettes wel geformateerd zijn. U kunt tijdens het gebruik vrij van diskette wisselen. Zolang het 'in werking' lampje (vlak onder de diskette-gleuf) niet brandt, kunt u de diskette verwisselen. Voor sommige funkties kan dit echter funest zijn!
Elke funktie kan onderbroken worden door BREAK. U keert dan terug naar het hoofdmenu.

De voor een BASIC-programmeur belangrijkste funkties worden hier behandeld.

Formateren van een diskette

Elke diskette moet voor gebruik geformateerd worden. Dit wil zeggen dat het magnetische oppervlakte van de diskette ingedeeld wordt in banen (tracks) en dat aan het begin van de diskette (voor de computer herkanbaar aan een gaatje in de diskette) een soort 'gebruiksaanwijzing bij de diskette' wordt gemaakt. Deze gebruiksaanwijzing is voor elk merk computer en voor elk type diskdrive iets anders. U kunt daardoor niet zomaar diskettes, die door een andere computer zijn aangemaakt, lezen (nog afgezien van het verschil in programmatuur tussen de verschillende merken).
De schijf met DOS (de zogenaamde 'master' diskette) hoeft niet geformateerd te worden, dit is al voor u in de fabriek gedaan. In het algemeen is het zelfs niet mogelijk deze diskette te formateren, doordat een kleine uitsparing rechtsboven in de diskette ontbreekt. Hetzelfde geldt voor commerciele software op diskette.

Het formateren van een diskette vernielt alle informatie die zich op de diskette bevindt. Wees dus voorzichtig met het formateren van een diskette als u niet zeker weet of er nog iets op staat (zie ook de volgende funktie).

Om een diskette te formateren, plaatst u de diskette in de diskdrive en kiest u van het DOS menu funktie I. FORMAT DISK. De computer vraagt u:

WHICH DRIVE TO FORMAT?–(welke drive formateren?)

Als u een diskdrive heeft zal dit altijd nummer 1 zijn. Heeft u meer dan een diskdrive, dan kunt u elk nummer intikken dat bij een aangesloten drive hoort. In de opgegeven drive moet de te formateren diskette zitten.
De computer geeft u nog een extra mogelijkheid om uzelf ervan te overtuigen dat u de DOS diskette uit de drive gehaald heeft en vervangen door de te formateren diskette:

TYPE "Y" TO FORMAT DISK nr (tik Y om disk nr te formateren)

Na het intikken van Y en RETURN gaat de diskdrive ongeveer een minuut lang rommelen, uw diskette is dan geformateerd en bruikbaar voor alle andere diskfunkties.

Bij DOS 3.0 gaat het formateren van een diskette op vergelijkbare wijze. Nu heet het echter initialiseren van een diskette. U kiest funktie I van Init. De computer laadt nu een hulpprogramma. Zodra dit gebeurd is, vraagt de computer u:

Format diskette in drive (1-8)?

Ook nu geeft u door een getal in te vullen weer aan welke diskdrive u wilt gebruiken. Bij een diskdrive is dat altijd nummer 1.

De volgende vraag is of u de diskette met enkele dichtheid of met dubbele dichtheid wilt beschrijven (single of dubble density). De dichtheid geeft aan hoeveel sporen er naast elkaar op de diskette komen te staan. Bij dubbele dichtheid staan er twee keer zoveel sporen naast elkaar. De sporen zelf zijn daardoor wel dunner.
U kiest een 1 als u een Atari 810 diskdrive heeft.
U kiest een 2 als u een Atari 1050 met dubble density diskettes gebruikt.

De computer vraagt nu of FMS.SYS geschreven moet worden. Dit bestand zorgt ervoor dat de communicatie tussen diskdrive en eventuele programmatuur gemakkelijker verloopt. Als u met bestanden gaat werken, is het handig om dit bestand op die diskettes te hebben, die u daarvoor gaat gebruiken. U hoeft dan dit bestand niet telkens vanuit DOS apart te laden. Antwoordt in dit geval met Y.
Het aanpassen van de FMS parameters is meestal niet nodig voor BASIC programmeurs. U kunt een N tikken.

Ook DOS 3.0 geeft u de mogelijkheid nog een keer te controleren of de juiste diskette wel in de drive zit. Daarnaast wordt gevraagd of u de juiste gegevens over density wel ingevuld heeft:
Druk op SHIFT-CLEAR als u toch nog iets wilt veranderen.
Controleer of de juiste diskette geplaatst is.
Druk op RETURN om het formateren te starten.

Met DOS 1.0, 2.0 en 2.5 kunt u ook een systeembestand op de diskette plaatsen. U doet dit met funktie H. WRITE DOS FILE.
DOS 1.0 en 2.5 zetten dan een enkel bestand op de diskette, DOS 2.0 plaatst twee bestanden. Bij DOS 2.0 en 2.5 kunt u kiezen naar welke drive het DOS bestand wordt geschreven. Bij DOS 1.0 wordt het bestand altijd naar drive 1 geschreven.

Een inhoudsopgave

Om een lijstje van alle op een diskette aanwezige bestanden en programma's te krijgen kiest u item A (1.0, 2.0 en 2.5) of de F van File index (3.0).
De computer vraagt u om nadere inlichtingen met:

```
DIRECTORY--SEARCH SPEC,LIST FILE?
```

U kunt op deze vraag antwoorden met het indrukken van RETURN. U krijgt dan een lijst met alle bestanden en programma's te zien.
Wilt u de inhoudsopgave van een diskette met programmatuur zien, dan kunt u voordat u antwoordt op de vraag van de computer, van diskette wisselen. Doet u dat niet, en zit de DOS diskette nog in de diskdrive dan krijgt u een lijst van bestanden op de DOS diskette te zien:

```
*DOS         SYS 039
*DUP         SYS 041
628 FREE SECTORS
```

Zo'n lijstje geeft veel informatie.
Het sterretje voor de naam geeft aan of het bestand al dan niet beschermd is (locked). Is dit het geval, dan kunt u niet zomaar over dat bestand heenschrijven, of het uitwissen.
DOS en DUP zijn in het voorbeeld hierboven de namen van de aanwezige bestanden.
SYS geeft een zogenaamde uitbreiding aan. U kunt elk bestand of programma van een uitbreiding voorzien. U doet dit door achter de naam een punt (.) en drie letters te plaatsen. Het is gebruikelijk om BASIC bestanden de uitbreiding .BAS te geven en tekstbestanden (van Atariwriter) de uitbreiding .TXT. Bijvoorbeeld:

```
LOAD "D:PROGRAMMA.BAS"
```

DOS 3.0 vraagt om iets andere gegevens:
Na het tikken van F, springt de computer onmiddellijk naar de volgende vraag: Filespec? Dit betekent dat de computer wil weten welk deel van de bestanden u wilt zien. Zie ook hiervoor onderstaande tekst. Om alle bestanden te zien drukt u op RETURN. De computer geeft dan een zogenaamd standaard-antwoord. Het antwoord dat hij verwacht dat u het meeste zult gebruiken, en al standaard in het programma heeft zitten: D1:*.* Dit wil zeggen: alle bestanden op de schijf in diskdrive 1. Druk nogmaals op RETURN.
Nu vraagt de computer waar U de inhoudsopgave wilt zien. Druk op RETURN om het standaardantwoord E: te zien. Druk nogmaals RETURN om de computer daadwerkelijk aan het werk te zetten.
De inhoudsopgave van de DOS 3.0 diskette is een stuk langer. Er bevinden zich allerlei hulpprogramma's op.

Gedeeltelijke inhoudsopgave

Als u een diskette hebt met erg veel programma's is het lang niet altijd handig om alle programma's en bestanden te moeten zien. Omdat de inhoudsopgave verdeeld is in verschillende kolommen, kunt u opdracht geven om alleen op een bepaalde kolom te letten. Belangrijk hierbij is het gebruik van de asterisk (*). Deze werkt als een soort 'joker'. Elk teken of groep van tekens kan hier ingevuld worden. Bijvoorbeeld:

*.BAS	Inhoudsopgave van alle bestanden met uitbreiding .BAS
*.TXT	Inhoudsopgave van alle bestanden met uitbreiding .TXT
PROG.*	Inhoudsopgave van alle bestanden met de naam PROG
A*.*	Inhoudsopgave van alle bestanden die beginnen met een A
A*.BAS	Inhoudsopgave van alle bestanden die beginnen met een A en als uitbreiding .BAS hebben.

Als u meer dan een diskdrive heeft, en u wilt een andere dan de eerste drive gebruiken, kunt u dat aangegeven door D2:, D3: of D4: voor de hierboven genoemde voorbeelden te plaatsen.

Inhoudsopgave op printer

In DOS 1.0, 2.0 en 2.5 kiest u een ander uitvoerapparaat dan het beeldscherm voor het weergeven van de inhoudsopgave, door achter het opgeven van de gewenste inhoudsopgave een komma te plaatsen met daarachter de code van het gewenste uitvoerapparaat. Bijvoorbeeld:

.,P:	Alles naar Printer
,P:	Idem
D2:*.BAS,P:	.BAS bestanden van drive 2 naar printer.

In DOS 3.0 kiest u het uitvoerapparaat door op de vraag 'Display device' te antwoorden met de lettercode voor het gewenste uitvoerapparaat.

Naar BASIC

Ongeacht of BASIC in de vorm van een cartridge aanwezig is, u verlaat DOS om naar BASIC over te schakelen door van het menu mogelijkheid B te kiezen (DOS 3.0: T van To cartridge). U bereikt hetzelfde effekt door de SYSTEM RESET knop in te drukken.

Kopieren van bestanden

Het kopieren van bestanden kunt u gebruiken om een reservekopie van een belangrijk bestand te maken. Met de COPY funktie is het mogelijk een kopie van een bestand op dezelfde diskette te maken, of op een andere diskette. Dit laatste gaat alleen als zowel de diskette met het origineel als de diskette waar de kopie op moet worden geplaatst, tegelijkertijd door de computer aangesproken kunnen worden. Dat betekent dat u twee diskdrives nodig heeft. Heeft u slechts een diskdrive, en wilt u toch van de ene diskette naar de andere diskette kopieren, dan moet u de DUPLICATE (dupliceer) funktie gebruiken.
Na het kiezen van C, vraagt de computer welk bestand u wilt kopieren, naar welk bestand:

```
COPY--FROM, TO
```

U geeft nu het originele bestand, een komma en het kopiebestand aan:

```
BESTAND1.BAS,BESTAND1.BAK
```

Zoals u ziet wordt de kopie aangegeven met de gebruikelijk uitbreiding .BAK. Dit staat voor BACK-UP (reserve). Na het indrukken van RETURN. gaat de diskdrive rommelen. Het bestand wordt gelezen. een nieuw bestand wordt geopend, en de inhoud van het origineel wordt gekopieerd. Als er al een bestand met de naam van het nieuwe bestand aanwezig is. dan wordt deze overschreven door het nieuwe bestand. De diskette mag niet door middel van een plakkertje tegen schrijven beschermd zijn. Het nieuwe bestand kunt u dan niet wegschrijven. Foutmelding 144 zal volgen. Het bestand DOS.SYS is beschermd. U kunt dit bestand niet met de COPY funktie kopieren (evenmin als de meeste commerciele software).
U kunt meer dan een bestand met dezelfde funktie kopieren. Daarvoor gebruikt u de joker (*) weer:

```
COPY--FROM, TO
*.BAS,D2:
```

Deze opdracht zal alle .BAS bestanden van de diskette in drive 1 kopieren naar de diskette in drive 2.

DOS 3.0 doet het kopieren weer iets anders. Zodra u de C van Copy ingetikt heeft, leest de computer een speciaal programma van de DOS schijf. De eerste vraag die na het laden verschijnt is:

```
Append (Y/N)? N
```

Het standaardantwoord is N(ee). Als u wel voor Append kiest, kunt u het ene bestand achter het andere bestand plakken. Kies dan antwoord Y(es). In DOS 1.0, 2.0 en 2.5 kunt u dit bereiken door achter de naam van het kopiebestand de combinatie /A te plaatsen. Op deze wijze wordt het originele bestand achter het 'kopie'bestand geplaatst. Het kopiebestand is daardoor geen kopie meer, maar bestaat dan uit een al bestaand bestand plus het originele bestand erachter geplakt.
Na de vraag over Append, vraagt DOS 3.0 om de lettercode van het apparaat waar het originele bestand vandaan moet komen. Door RETURN in te drukken verschijnt het standaardantwoord D1:
Door weer op RETURN te drukken, verschijnt de vraag om de bestandsnaam. Vul hier de naam in van het originele bestand en druk op RETURN.
De vraag Destination device? slaat op het uitvoerapparaat. Het standaardantwoord D1: verschijnt door het indrukken van RETURN.
Nu kunt u de naam van het kopiebestand opgeven. Tot slot wordt u gevraagd of zowel het originele bestand als het kopiebestand op dezelfde diskette staan. Het standaardantwoord is N.

Niet iedereen heeft twee diskdrives, terwijl het voor bijna iedereen belangrijk is om van bestanden reserve exemplaren te hebben.
U dient funktie O te gebruiken om met een enkele diskdrive, bestanden van de ene diskette naar de andere te kopieren.
De computer vraagt u welk bestand gekopieerd moet worden:

NAME OF FILE TO MOVE

U kunt nu de naam van het te kopieren bestand opgeven. U mag gebruik maken van een of meer asterisken (jokers) om verschillende bestanden te kopieren. U hoeft geen diskdrivenummer op te geven.
Bij deze funktie wordt alleen gebruik gemaakt van de eerste drive. Met de asterisk kunt u geen bestanden aangegeven met .SYS, kopieren. Deze dient u expliciet op te geven. De computer geeft vervolgens telkens aan of u de diskette met het origineel (de brondiskette) of de diskette waar de kopie naar toe moet (de bestemmingsdiskette), in de diskdrive moet stoppen. Telkens na het wisselen van een diskette moet u op RETURN drukken.
Om vergissingen hierbij te voorkomen is het raadzaam de brondiskette te beschermen tegen foutief overschrijven door afplakken van de uitsparing aan de zijkant van de diskette met een stickertje.

U kunt de hele schijf (met uitzondering van de .SYS) kopieren met *.*, of met de special funktie J. DUPLICATE DISK. Deze funktie kopieert ook de .SYS bestanden.
U kunt met funktie J zowel van de ene drive naar de andere drive kopieren, als kopieren met een enkele drive. In het eerste geval geeft u op de vraag:

DUP DISK-SOURCE,DEST DRIVES? (dupl. diskette, bron, bestemming)

als antwoord de nummers van de te gebruiken diskdrives. Bijvoorbeeld: 1,2
Als u een diskdrive heeft, geeft u als antwoord: 1,1. De computer zal u dan telkens vragen ofwel de brondiskette ofwel de bestemmingsdiskette in de diskdrive te doen.

De DUPLICATE funktie (D) van DOS 3.0 werkt vergelijkbaar. Informatie wordt van de brondiskette op de bestemmingsdiskette geplaatst.
De bestemmingsdiskette wordt, indien nodig, eerst geformateerd.
De duplicatefunktie wordt door DOS 3.0 van diskette gelezen. De DOS diskette moet zich in de diskdrive bevinden als u deze funktie van het menu kiest.
De computer begint met een speciaal DUPDISK programma van de diskette te laden. De eerste vraag die u krijgt is:

Source drive number?—(bron drive nummer)

Tik hier het nummer van de drive in, waar de brondiskette zich bevindt. Als u een diskdrive heeft, is dit altijd drive 1.
Vervolgens vraagt de computer u de drive met de bestemmingsdiskette op te geven. Als u een diskdrive heeft, is dat wederom drive 1. Bij twee of meer drives, vraagt de computer u de diskettes in de drives te plaatsen en op RETURN te drukken. Bij een diskdrive moet u op aanwijzingen van de computer de diskettes telkens wisselen en op RETURN drukken. Plak de brondiskette af!

Verwijderen van overbodige bestanden

Soms zijn bestanden niet meer nodig. U heeft een nieuw, beter of langer bestand gemaakt of u heeft een programmeerproject overboord gezet. Overbodige bestanden nemen alleen maar ruimte in beslag op de diskette, en ze maken de inhoudsopgave van de diskette minder leesbaar.
U kiest de funktie D. DELETE FILES(S). De computer vraagt u welk bestand u wilt verwijderen:

DELETE FILE SPEC

U dient een drive en een of meer bestanden op te geven. U kunt asterisken gebruiken. Bijvoorbeeld:

D1:TEST.*

Om alle bestanden met de naam TEST te verwijderen van de diskette in drive 1 (standaard als er maar een drive is).
De computer gaat op zoek naar het opgegeven bestand en geeft ondertussen weer:

TYPE "Y" TO DELETE...-

Nu worden alle bestanden die in aanmerking komen om gewist te worden opgesomd. Telkens als u het ermee eens bent dat het opgegeven bestand gewist wordt, tikt u Y gevolgd door RETURN.
Een bestand dat op slot zit, kan niet gewist worden.
U kunt alle bestanden wissen door *.* op te geven.
Als u niet achter elke bestandsnaam wilt opgeven of deze gewist moet worden, kunt u achter de naamopgave de tekens /N zetten. De vraag TYPE "Y" TO DELETE... wordt dan weggelaten. De opgegeven bestanden worden automatisch gewist.
./N wist alle bestanden van een schijf zonder om bevestiging te vragen. Wees voorzichtig met het wissen van bestanden. Een eenmaal gewist bestand kan niet meer worden gered.

Het wissen van een bestand met DOS 3.0 heet ERASE (E).
De computer vraagt om de bestandspecificaties van het te wissen bestand. U tikt bijvoorbeeld:

D1:*.*

Hiermee kunt u alle bestanden op de diskette in drive 1 verwijderen.
De computer geeft alle er zake doende bestanden weer en vraagt: Erase all specified files (Y/N)? (wis alle aangegeven bestanden (ja/nee)?). Als u alle opgegeven bestanden kwijt wilt, tikt u Y. De computer vertrouwt u echter niet erg, en vraagt: Are you sure (Y/N)? (Zeker weten?). Pas na uw Y en RETURN worden de bestanden gewist.

Bestanden beschermen

Een ongeluk zit in een klein schijfje. Voordat u het weet heeft u een van uw kostbare bestanden of programma's gewist of overgeschreven door een nieuw programma of bestand. Om dit te voorkomen kunt u uw bestanden beschermen door ze 'op slot' te doen. Dat wil zeggen dat de computer zo'n programma niet kan wissen of overschrijven. Wilt u dat toch doen, dan moet het bestand eerst van slot worden gedaan. Dit doet u met de funktie F. LOCK FILES. De computer vraagt:

WHAT FILE TO LOCK?-(Welk bestand op slot doen?)

U kunt nu (eventueel) een drivenummer en de naam van het bestand opgeven. U kunt gebruik maken van asterisken om meer dan een bestand tegelijk op slot te doen.

DOS 3.0 doet bestanden op slot door PROTECT (bescherm). Ook na het kiezen van deze funktie kunt u drivenummer en naam met eventueel asterisken opgeven. U moet uw keuze bevestigen met Y RETURN.

Het op slot doen van bestanden beschermt ze niet tegen het opnieuw formateren van de diskette. Ook de beschermde bestanden worden dan gewist. Hetzelfde geldt voor het kopieren van een complete diskette door de DUPLICATE funktie van DOS 3.0. Deze houdt mede het initialiseren van de diskette in.

Bestanden van slot doen

Het omgekeerde van bovenstaande funktie. In DOS 1.0, 2.0 en 2.5 kiest u G. UNLOCK FILES.
In DOS 3.0 kiest u voor UNPROTECT.
Zie verder bij het beschermen van bestanden.

Veranderen van bestandsnamen

U kunt de naam van een bestand veranderen door E. RENAME FILE.
U ziet de vraag:

RENAME - GIVE OLD NAME, NEW

U geeft nu de oude naam op, eventueel met een drivespecificatie, een komma en tot slot de nieuwe naam. Bij de nieuwe naam mag u geen drive specificeren. De computer gebruikt dezelfde drive voor het bestand met de oude naam en het bestand met de nieuwe naam. Voor het overbrengen van een bestand van de ene diskette naar de andere dient u C. COPY FILE te gebruiken.

Als de nieuwe naam al bestaat, krijgt u twee bestanden met dezelfe naam. Deze zullen allebei reageren op die nieuwe naam. U kunt niet een van de twee aanroepen, of een van de twee een nieuwe naam geven. De enige remedie is het verwijderen van beide bestanden en opnieuw beginnen.
U mag gebruik maken van asterisken, maar wees hier uiterst voorzichtig mee. Voor u het weet heeft u twee bestanden met dezelfde naam.

In DOS 3.0 hoeft u niet bang te zijn voor een dubbele bestandsnaam. Als dit dreigt te gebeuren geeft de computer een foutmelding.
U krijgt de funktie van herbenoemen in DOS 3.0 door RENAME te kiezen. De computer zal u vragen de oude en vervolgens de nieuwe bestandsnaam op te geven.

Beschermen van BASIC programma's

Zowel DOS 2.0, 2.5 als 3.0 kennen een speciaal programma om ervoor te zorgen dat uw BASIC programma's in het geheugen van de computer niet verloren gaan door gebruik te maken van DOS funkties.
Normaal gesproken gebruiken DOS funkties ook dat deel van het geheugen van de computer waar uw BASIC programma staat. In dat geval wordt het BASIC programma vernietigd.
Door een speciale MEM.SAVE op uw diskette te hebben staan, wordt het geheugendeel waar het BASIC programma staat eerst op de diskette geplaatst, vervolgens de DOS funktie verricht, en als u weer terugkeert naar BASIC, de inhoud van het geheugen weer gevuld met de informatie die tijdelijk op de diskette opgeslagen was.
Telkens als u DOS intikt, wordt MEM.SAVE aangeroepen. Telkens als u naar BASIC gaat, wordt de inhoud van MEM.SAVE weer in het geheugen geplaatst.
Nadeel van MEM.SAVE is dat het heen er weer plaatsen van de inhoud van het geheugen natuurlijk wel wat tijd in beslag neemt. Het overschakelen van DOS naar BASIC en vice versa gaat een stuk langzamer.
DOS wordt door DOS 2.0 en 2.5 geladen vanuit het bestand DUP.SYS. Elke diskette met DUP.SYS kan gebruikt worden door de DOS opdracht. Om MEM.SAVE te gebruiken, moet deze op dezelfde diskette staan.

Om MEM.SAVE op een diskette te plaatsen kiest u uit het DOS menu: N. CREATE MEM.SAVE.
U hoeft vervolgens alleen de gewenste diskette in de drive te doen, en Y RETURN in te drukken.

BASIC programma's

Atari BASIC kent vijf statements die geschikt zijn voor gebruik met complete BASIC programma's die op disk (moeten komen te) staan.

SAVE
Deze opdracht plaatst een BASIC programma in het geheugen op een diskette in de aangegeven drive, onder de aangegeven naam. Het programma wordt in gecodereerde vorm opgeslagen.

```
SAVE "D1:PROGRAMMA.BAS"
```

U kunt een nummer achter de D weglaten. De D zelf is verplicht.

LIST
Deze opdracht is gelijk aan SAVE. Nu wordt het programma niet gecodeerd opgeslagen, maar als een serie ASCII codes. Voor elk teken in de programma-listing wordt een ACCII code op de diskette geplaatst.

```
LIST "D1:PROGRAMMA.LST"
```

Met de LIST-opdracht kunnen ook gedeelten van een programma worden opgeslagen. Daarvoor dient u een begin- en einderegelnummer op te geven.

LOAD
Deze opdracht laadt een programma van de diskette in het geheugen. Alleen programma's die door middel van SAVE zijn opgeslagen, kunnen door middel van LOAD worden geladen.

```
LOAD "D2:PROG.BAS"
```

De LOAD opdracht wist een eventueel in het geheugen aanwezig programma uit. Het nieuwe programma vervangt het oude.

ENTER
Idem als LOAD, maar voor programma's die opgeslagen zijn door middel van LIST.

```
ENTER "D1:PROGRAMMA.LST"
```

De ENTER opdracht wist een oud programma in het geheugen niet uit. Alleen de regels die in het oude en nieuwe programma dezelfde regelnummers

hebben, worden vervangen door de regels uit het nieuwe programma. Om het oude programma te wissen, dient u NEW te gebruiken, voordat u ENTER gebruikt.

RUN
Een van de eerste opdrachten die u geeft als u een nieuw programma in het geheugen geladen heeft, is de RUN opdracht. U moet een programma eerst laden en dan runnen. U kunt deze twee stappen samenvoegen door achter de RUN opdracht de naam van het programma te plaatsen. In dat geval omvat de RUN opdracht tevens de LOAD opdracht.

```
RUN "D1:PROGRAMMA.BAS"
```

U kunt deze vorm van een RUN opdracht ook in een programma gebruiken. Daarmee kunt u een programma laden en runnen vanuit een ander programma. U kunt dit afhankelijk van een bepaalde voorwaarde maken door een IF..THEN.. opdracht:

```
IF X < 0 THEN RUN "D:NULP.BAS"
```

De opdracht RUN wist alle variabelen in het geheugen. Als u een programma oproept in een ander programma, bent u de waarden van de variabelen kwijt.

Bestanden op diskette

Voor het werken met bestanden en de daarvoor nodige BASIC opdrachten u verwezen naar hoodstuk 12 Bestanden verwerken.

BIJLAGE 7: EXTRA GEHEUGEN VAN DE 130 XE

De Atari 130 XE heeft veel meer geheugenruimte ter beschikking dan de 65XE of de 800 XL. Om precies te zijn: de Atari 130XE heeft 131.072 bytes aan geheugenruimte.
Het zal niet lang duren of allerlei commerciele software zal deze overvloed aan geheugenruimte gaan gebruiken. Een uiterst nuttige toepassing van deze extra geheugenruimte is tekstverwerking. De schrijver kan met zoveel geheugenruimte veel meer tekst in een bestand plaatsen, waardoor grotere stukken tekst gemanipuleerd kunnen worden.
Maar ook de BASIC programmeur kan goed gebruik maken van extra geheugenruimte. Zelfs als uw programma's nog niet zo lang zijn, dat u extra geheugenruimte nodig heeft, dan nog is zoveel geheugenruimte zeer handig. Het is mogelijk om een deel van het geheugen als een 'disk-drive' te laten werken. Zo'n 'RAM-disk' is vele malen sneller in het opnemen en weer afgeven van gegevens dan een normale diskdrive. Dat betekent dat u, als u de computer net aanzet, een gedeelte van een diskette overzet naar de RAM-disk, en vervolgens snel kunt werken met de RAM-disk. Vlak voordat u de computer weer uit gaat zetten, kopieert u het volledige geheugen van de RAM-disk weer naar diskette. Een enorme tijdsbesparing tijdens het werken, met daarbij de mogelijkheid om zo snel gegevens met de RAM-disk uit te wisselen, dat bijvoorbeeld animatie mogelijk is.

Uw DOS 2.5 master diskette bevat een bestand dat RAMDISK.COM heet. Dit bestand zorgt er automatisch voor dat uw RAMdisk wordt ingesteld.
Als u DOS laadt, zorgt de computer ervoor dat:
- de RAMdisk wordt ingesteld.
- de 64 K extra geheugenruimte als diskdrive nummer 8 wordt beschouwd.
- DUP.SYS en MEM.SAV in het geheugen worden gekopieerd en indien nodig worden gebruikt vanuit het geheugen in plaats vanaf de diskette.

De RAMdisk is niet zo groot dat er een complete diskette in past. U kunt dus niet gebruik maken van de funktie J. DUPLICATE DISK om een diskette naar het geheugen te kopieren. U kunt wel individuele bestanden kopieren. Als u de computer uit gaat zetten, kunt u natuurlijk J.DUPLICATE wel gebruiken om de inhoud van de RAMdisk op diskette te zetten.

Als u de RAMdisk niet wilt gebruiken dient u het bestand RAMDISK.COM buiten werking te stellen. De beste methode hiervoor is een kopie te maken van de gehele DOS 2.5 master diskette en van de kopie het bestand RAMDISK.COM te verwijderen. De kopie merkt u nu met een label: MASTER ZONDER RAMDISK. De originele master merkt u nu met een label: MASTER MET RAMDISK. Zo heeft u twee master diskettes en kunt u naar behoefte werken met of zonder RAMdisk.

BASIC en extra geheugenruimte

De twee centrale chips in de Atari: de 6502 microprocessor en de ANTIC chip (voor het videogedeelte) kunnen elk slechts 65.536 bytes besturen. Met de Atari 130XE heeft u echter het dubbele aan geheugenruimte. Het is mogelijk de computer opdracht te geven om naar een van de twee blokken te kijken. Door tussen de twee blokken te wisselen, het zogenaamde 'bank-switching', kunt u effectief over alle geheugenruimte beschikken.
U kunt de extra geheugenruimte niet tegelijk voor uw BASIC programma's en als RAMdisk gebruiken.
Bank-switching schakelt (tijdelijk) een 16 K geheugengebied af, en gebruikt daarvoor in de plaats een ander 16 K geheugengebied.
In de 130XE vindt u de extra geheugenruimte in de geheugenlokaties 16384 tot en met 32767. Door een wijzer naar een bepaald blok van 16 K te laten wijzen, kan de computer gedwongen worden om dat deel van het geheugen te gebruiken.
De wijzer die daarvoor dient te worden gebruikt, bevindt zich in geheugenlokatie 54017. De vierde en vijfde bit van deze geheugenlokatie bepalen welk van de chips gebruik zal maken van het extra blok geheugenruimte. Beide bits staan zonder een POOK opdracht van de gebruiker op 1. Bit 4 bepaalt wat de CPU (de 6502) doet en bit 5 bepaalt wat de ANTIC doet. Door bit 4 in te stellen op 0, gaat de 6502 een blok uit de extra geheugenruimte gebruiken. Door bit 5 in te stellen op 0, gaat de ANTIC een blok uit de extra geheugenruimte gebruiken. Door zowel bit 4 als bit 5 op 0 in te stellen, gebruiken beide chips een blok uit de extra geheugenruimte. Welk blok van 16 K uit de extra geheugenruimte wordt gebruikt, is afhankelijk van de instelling van de tweede en derde bit van geheugenlokatie 54017.
Er zijn vier blokken van 16 K beschikbaar in het extra geheugengebied. Met twee bits (bis 2 en 3) die elk op 0 of 1 kunnen staan, zijn vier combinaties mogelijk. Elke combinatie wijst dus een van de 16 K blokken aan.
U hoeft niet zelf ingewikkeld aan het rekenen te slaan om uit te rekenen welke waarde u in geheugenlokatie 54017 moet POKEn om het juiste blok aan de juiste chip toe te wijzen (wilt u toch gaan rekenen, zie dan ook bijlage 9: Binair/decimaal getalstelsel).
De formule om een waarde te bepalen voor de wijzer van lokatie 54017 is:

```
POKE 54017,193 + 4 * blok + 16 * chip
```

Voor het gemak zijn de vier 16 K blokken genummerd van 0 tot en met 3. Deze nummering slaat op de volgende adressen in het extra geheugen:

BLOK	Adres in extra geheugen
0	0 tot en met 16383
1	16384 tot en met 32767
2	32768 tot en met 49151
3	49152 tot en met 65535

Er zijn vier combinaties mogelijk voor het gebruik van het blok door de 6502 of door ANTIC. Deze vier mogelijkheden zijn in de formule met 'chip' aangegeven en genummerd van 0 tot en met 3:

Chip	Gebruik normale of extra geheugenruimte door:	
	6502	ANTIC
0	extra	extra
1	normaal	extra
2	extra	normaal
3	normaal	normaal

Vergeet u niet dat de minimumwaarde van lokatie 54017 altijd 193 moet zijn.

BIJLAGE 8: TOETSENBORD CODES

Elke ingedrukte toets zorgt voor een eigen code die de Atari intern gebruikt. Deze code is niet gelijk aan de ASCII code.

Toets	Code
A	63
B	21
C	18
D	58
E	42
F	56
G	61
H	57
I	13
J	1
K	5
L	0
M	37
N	35
O	8
P	10
Q	47
R	40
S	62
T	45
U	11
V	16
W	46
X	22
Y	43

Toets	Code
Z	23
0	50
1	31
2	30
3	26
4	24
5	29
6	27
7	51
8	53
9	48
:	2
+	6
*	7
=	15
<	54
>	55
ESC	28
RETURN-1	
SPC	33
INV	39
TAB	44
DEL	52
CAP	60

TOETS + SHIFT	CODE
A	127
B	85
C	82
D	122
E	106
F	120
G	125
H	121
I	77
J	65
K	70
L	64
M	101
N	99
O	72
P	74
Q	111
R	104

TOETS + SHIFT	CODE
S	126
T	109
U	75
V	80
W	110
X	86
Y	107
Z	87
0	114
1	95
2	94
3	90
4	88
5	93
6	91
7	115
8	117
9	112

BIJLAGE 9: BINAIR/DECIMAAL GETALSTELSEL

Getalstelsels

De mens rekent meestal in een tientallig stelsel. Dat wil zeggen dat hij van 0 tot en met 9 telt en eventueel weer opnieuw bij 0 begint, daarbij onthoudend hoe vaak hij al tot negen geteld heeft.
Toch is rekenen met andere soorten getallen niet onmogelijk. De meeste lezers zullen het Engelse geldsysteem van vroeger nog wel kennen. Heel moeilijk voor iemand die gewend is aan een decimaal (=tientallig) geldstelsel, maar voor Engelsen geen enkel probleem. Het Engelse en Amerikaanse systeem voor het meten en weergeven van lengtematen (yards, feet en inches) is ook niet tientallig, maar is voor de dagelijkse gebruiker niet moeilijk.
Een computer werkt zelf ook niet decimaal. Een computer bestaat uit heel veel kleine schakelaartjes. Elke schakelaar kan of aan of uit staan. Daarom kan een computer slechts tot een tellen: 0 en 1. Daarna begint en computer opnieuw bij 0 waarbij onthouden wordt hoe vaak er al tot 1 geteld is. Deze manier van tellen wordt het twee-tallige of binaire getalsysteem genoemd.
Het lijkt onzinnig om ons te verdiepen in de interne telwijze van de computer als het resultaat toch altijd in een voor de mens eenvoudig te begrijpen tientallige vorm op het scherm komt te staan. In de hoofdstukken over grafiek en kleur zult u al snel merken dat het erg handig is als u weet hoe de computer intern allerlei gegevens opslaat. Als u later in machinetaal gaat programmeren dan is deze kennis zelfs onontbeerlijk.
Een Atari-computer is een 8-bits computer. Dat betekent dat de geheugenbits gegroepeerd zijn in groepen van 8-bits. Zo'n groep heet een byte. Binnen zo'n groep van 8 bits kan geteld worden tot het getal 11111111 (binair). Dat is gelijk aan 255 decimaal. Omdat elk binair cijfer staat voor een telling van 0 tot en met 1, kunnen binaire getallen omgezet worden in decimale door gebruik te maken van machten van 2.

Binair	Decimaal	Binair	Decimaal
00000000	0	00001011	11
00000001	1	00001100	12
00000010	2	00001101	13
00000011	3	00001110	14
00000100	4	00001111	15
00000101	5	00010000	16
00000110	6	00010001	17
00000111	7	00010010	18
00001000	8	00010011	19
00001001	9	00010100	20
00001010	10	00010101	21

Het omzetten van een binair getal naar een decimaal getal gaat als volgt:

```
1 0 0 1 0 1 0 1
|
1 * 2^0 = 1 *   1 =   1
0 * 2^1 = 0 *   2 =   0
1 * 2^2 = 1 *   4 =   4
0 * 2^3 = 0 *   8 =   0
1 * 2^4 = 1 *  16 =  16
0 * 2^5 = 0 *  32 =   0
0 * 2^6 = 0 *  64 =   0
1 * 2^7 = 1 * 128 = 128   +
                    ---------
10010101 binair =   149 decimaal
```

Let op! Het tellen van de machten van twee begint rechts. De kleinste macht staat het meeste naar rechts. Dat geldt immers ook voor decimale cijfers: het kleinste cijfer staat het meeste naar rechts (de eentallen, links daarvan de tientallen, links daarvan de honderdtallen, links daarvan de duizendtallen, etc.)

U ziet dat het omzetten met machten van twee gaat. Andersom dient u zo groot mogelijke machten van twee van het getal af te trekken. Om 95 decimaal om te zetten in een binair getal gaat u als volgt te werk:

95	aftrekken 2^7 ?	lukt niet	bit 8 = 0
95	aftrekken 2^6 ?	lukt (rest 31)	bit 7 = 1
31	aftrekken 2^5 ?	lukt niet	bit 6 = 0
31	aftrekken 2^4 ?	lukt (rest 15)	bit 5 = 1
15	aftrekken 2^3 ?	lukt (rest 7)	bit 4 = 1
7	aftrekken 2^2 ?	lukt (rest 3)	bit 3 = 1
3	aftrekken 2^1 ?	lukt (rest 1)	bit 2 = 1
1	aftrekken 2^0 ?	lukt (rest 0)	bit 1 = 1

95 decimaal = 01011111 binair.

Let op dat bit 8 de meest linkse bit is, en bit 1 de meest rechtse bit. Zie hiervoor.

INDEX

aanhalingstekens 18
aansluitingen 13
aanval-fase 241
aapje 183
ABS 164,165
AND 75
apparaatnummer 305
arrays 97,98,99,102
arrays (strings) 100,101
arrays (twee-dimensionaal) 99
ASC 85
ASCII-codes 272-276
assenstelsel 188,299
ATASCII-codes 272-276
ATN 212
audio-visual test 17
auto-repeat 25

BASIC 9,10
BASIC statements 283
beeldteller 67,68
beeldweergave 55,56
bestanden 112,113,117,119,120,121,130
bestanden (cassette) 112,113,117,119,120,121,130
bestanden (diskette) 112,113,116,119
BREAK-toets 23
bubblesort 107

CAPS-toets 19
cassette 112
CHR$ 85
cirkels 211
CONTROL-toets 19
coordinaten 253
CLOAD 42
CLR 88
COLOR 154,188
COS 211
CSAVE 41
cursor aan/uit 168

DATA 93
datalijsten 103
DEG 250
DELETE-toets 21,22
dieptesuggestie 21
DIM 82,97
dimensies 247
direkte opdrachten 29
disk-bewerkingen 335
disk-drive 26,333
diskette 112,335
DRAWTO 154,187,207

ESCAPE-toets 23
END 135
ENTER "C:" 43
ENTER "D:" 43
extra geheugen 130XE 347

faculteit 71,72
FOR..NEXT.. 64
fouten verbeteren 30,31,32
foutmelding 116,119,127,277
FRE 81
frequentie 235

geluid 235
geluidseffecten 245
generator (geluid) 235
getalstelsel 353
GET# 120
giraf 225,226,227
golfvorm 235
GOSUB 135
GOTO 73-75
grafiek 167,187,205,219
grafiek 3-dimensionaal 253
grafiek, player-missile 219
grafische symbolen 146
GRAPHICS 59,60,141
GTIA 159

hang-fase 241
HELP-toets 29
hoge resolutie 205

IF..THEN.. 73
inkleuren 214
I/O 13
INPUT 53

INPUT# 123
INSERT-toets 21
INT 78
interpreter 12
integers 50,78
INVERSE-toets 24
invoer 53
invoer/uitvoer 13
IOCB 12, 114,115,327
IOCB taken 115

joystick 14,195,198

keyboard-test 17
kleuren 147,148,151,154
kleurregister 154,155
klok 66,67
kolom 153,156
krommen 207
kubus 259

leestekens 56
LEN 90
LET 49,50
letterteken 174,177
lettertoetsen 18
lijnen 206
lijsten 101,97
LIST 36
LIST "C:" 42
LIST "D:" 43
LOAD "D:" 43
logica 73,76
loslaat-fase 241
lussen 63,66,68
lussen, genestelde 70,71

matrices 258
memory-test 16
microcomputer 11
microprocessor 11

NEW 39
NEXT 64
NOT 75,76
notitieboek 91

omhullende 240
ON..GOSUB.. 137
ON..GOTO.. 138
OPEN# 115
operators 75-77
oplossend vermogen 142,143
opmaak-toetsen 21
OPTION-toets 24
OR 75

palindromen 91
PEEK 67,323
PI 212
piano 237
pijl-toetsen 25
player-missile grafiek 219
PLOT 154,188
PM-grafiek 225
PM-grafiek (kleuren) 232,233
PM-grafiek (voorrang) 230,231
POKE 58,323
POSITION 60,61,151,154,215
PRINT 29
PRINT# 121
PRINT#6 60,149
printer 14
programma 9,35-37
projectie 255
projectiel 221
PUT# 117

RAM 12
randapparatuur 14
raster 142,143
READ 93
reele variabelen 50
regelnummers 35,36
REM 40,134
RESET-toets 24
resolutie 148
RESTORE 95
RETURN 135
RETURN-toets 23
rij 151,154
RND 78
ROM 12
rotatie 21
RUN 35

SAVE"D:" 43
schaduwwerking 248
schermen 141
schermopmaak 25
scherm wissen 34
SELECT-toets 24
SETCOLOR 147,164
SHIFT-toets 18,19
SIN 211
sinussen 190,213
soortaanduiding 44,122
sortezen 105-108
SOUND 236
speler 219
SQR 165
START-toets 29
STICK 199
stippen 205
STRIG 201
stringarrays 100,101
strings 81,82
strings (manipulatie van) 89
strings (rekenen met) 84
stringvariabelen 83
stroomdiagram 38,39
stuiterbal 208
subroutines 133

taartgrafiek 191
TAB-toets 25
tabulatie 57
tekstvenster 142
THEN 73,74
tijd 66-68
toetsenbord 18,195
toetsenbordcodes 351
toetsenbord-test 17
TRAP 127,128
tunnels 250

VAL 87
variabelen 47-50
vector 256
vergelijkingen 73
verval-fase 241,242
volume 236
voortschrijdend gemiddelde 193

wild-card 44
wisselroutine 105-107

X-as 258
XIO 215,330

Y-as 258

Z-as 258
zelf-maak-tekens 174